Entwurf und Implementierung eines Expeditionsportals für eine wissenschaftliche Großforschungseinrichtung

Diplomarbeit

zur Erlangung des Grades Diplom Informatikerin (FH)

an der
Hochschule Bremerhaven
Fachbereich Informatik/Wirtschaftsinformatik

Studiengang mit Schwerpunkt Informatik

vorgelegt von: Karen Mindermann
Matr.-Nr.: 21202

aus: An der Allee 36
27568 Bremerhaven
Tel.: 0471-4833702
Kmindermann@awi-bremerhaven.de

Referent: Prof. Dr. Günter Matthiessen
Korreferentin: Dr. Ana Macario

Bremerhaven, den 29.09.2003

Danksagung

An dieser Stelle möchte ich allen danken, die mich bei der Erstellung dieser Arbeit fachlich und persönlich unterstützt und begleitet haben.

Ich danke Prof. Dr. Günter Matthiessen und Dr. Ana Macario für die Betreuung meiner Diplomarbeit.

Besonderer Dank geht an Verena Graßmann, die immer mit Rat und Tat beiseite stand.

Außerdem danke ich den Mitarbeitern des AWI, insbesondere dem Studentenraum, dem Nummerngirl und Sebastian Lorr für Motivation, Unterstützung und ein gutes Arbeitsklima.

Außerdem danke ich meinen Eltern und meiner Familie, die mir das Studium ermöglicht haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACI	Access Control Information
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASP	Active Server Pages
AWI	Alfred-Wegener-Institut für Polar- und Meeresforschung
CGI	Common Gateway Interface
DB	Database
DIT	Directory Information Tree
DN	Distinguished Name
DS	Directory Server
E/A	Ein-/Ausgabe
FTP	File Transfer Protocol
GB	Gigabyte
GUI	Graphical User Interface
HGF	Hermann von Helmholtz-Gemeinschaft Deutscher Forschungszentren
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IEFT	Internet Engineering Task Force
IP	Internet Protocol
JDK	Java Development Kit
JNDI	Java Naming and Directory Interface
JSP	Java Server Pages
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
OOA	Objectoriented Analysis
OOD	Objectoriented Design
PDF	Portable Document Format
PHP	HyperText Preprocessor
RAM	Random Access Memory
RDN	Relative Distinguished Name
SASL	Simple Authentication and Security Layer
SDK	Software Development Kit

SSL	Secure Socket Layer
TCP	Transfer Control Protocol
UI	User Interface
UID	Unique Identifier
UML	Unified Modeling Language
WWW	World Wide Web

1 Einleitung

Das Alfred-Wegener-Institut für Polar- und Meeresforschung (AWI) ist Mitglied der Hermann von Helmholtz-Gemeinschaft Deutscher Forschungszentren (HGF). Das Institut wurde 1980 in Bremerhaven als Stiftung des öffentlichen Rechts gegründet. Die Stiftung Alfred-Wegener-Institut für Polar- und Meeresforschung umfasst das Alfred-Wegener-Institut für Polar- und Meeresforschung in Bremerhaven, die Forschungsstelle Potsdam (1992), die Biologische Anstalt Helgoland und die Wattenmeerstation Sylt.
Die Stiftung Alfred-Wegener-Institut führt wissenschaftliche Projekte in der Arktis, Antarktis und den gemäßigten Breiten durch. Sie koordiniert die Polarforschung in Deutschland und stellt die für Polarexpeditionen erforderliche Ausrüstung und Logistik zur Verfügung. Ziel der wissenschaftlichen Arbeit ist ein besseres Verständnis der Beziehungen zwischen Ozean, Eis und Atmosphäre, der Tier- und Pflanzenwelt der Arktis und Antarktis sowie der Entwicklungsgeschichte der polaren Kontinente und Meere. Das AWI arbeitet in zahlreichen internationalen Forschungsprogrammen und steht in engem Kontakt mit zahlreichen Universitäten und Institutionen in Europa und Übersee. Es entsendet Wissenschaftler an Institute in aller Welt, auf andere Forschungsschiffe und Stationen und lädt Wissenschaftler anderer Nationen auf die "Polarstern" und nach Bremerhaven und Potsdam ein. Etwa ein Viertel der Teilnehmer an "Polarstern“-Expeditionen sind ausländische Wissenschaftler. [1]

1.1 Aufgabenstellung

Im Rahmen dieser Diplomarbeit soll ein Expeditionsportal als Anwendung für das Intranet und die Internetseite des Alfred-Wegener-Instituts entwickelt werden. Gegenstand des Expeditionsportals ist die Repräsentation der am AWI durchgeführten Expeditionsfahrten der Forschungsschiffe und der AWI Forschungseinrichtungen wie z.B. der Forschungsschiffe oder der Forschungsstationen. Daneben soll die Anwendung auch eine Administrationsfunktion bieten, die es möglich macht, die eingetragenen Expeditionsfahrten zu verwalten. Dies beinhaltet Funktionen zum Löschen, Modifizieren und Anlegen neuer Expeditionsfahrten.

1.2 Ziel der Arbeit

Diese Arbeit stellt die schriftliche Dokumentation des zu entwickelnden Prototyps dar.

[1] vgl. [AWI, 2003]

Sie dient der Erlangung des Grades einer Diplom Informatikerin. Gleichfalls ist sie Dokumentation für das AWI, anhand derer die Entwicklung und die Implementierung der Anwendung nachvollzogen werden kann.

1.3 Aufbau der Arbeit

Die vorliegende Arbeit beschreibt den Entwurf und die Implementierung des zu entwickelnden Prototyps des Expeditionsportals.

Nach dieser Einleitung folgt im Kapitel 2 die Beschreibung der Ist-Situation. In Kapitel 3 werden die Anforderungen an das zu entwickelnde System analysiert und beschrieben. Die Kapitel 4-6 beinhalten Beschreibungen zu den theoretischen Hintergründen und technischen Rahmenbedingungen des Systems. Als Vorgehensmodell für die Software-Entwicklung wurden die Objektorientierte Analyse (OOA) und das Objektorientierte Design (OOD) nach Heide Balzert gewählt. Dabei wird zur Visualisierung die Notation der Unified Modeling Language (UML) eingesetzt.[2]

Die Dokumentation dieser Analyse- und Entwurfsschritte erfolgt in den Kapiteln 7-8. In Kapitel 8 wird die Implementierung des Systems beschrieben. Kapitel 10 bildet mit einem Ausblick auf die zukünftige Nutzung und Erweiterungsmöglichkeiten der Anwendung und einem Resümee den Abschluss dieser Arbeit.

[2] vgl. [Balzert, 1999]

2 Ist-Situation

In der Ist-Analyse wird das bereits vorhandene System analysiert und eventuelle Schwachstellen werden aufgezeigt. Durch die Analyse der Ist-Situation können sich neue Anforderungen für das zu entwickelnde System ergeben.

Im folgenden wird die bestehende Situation in Bezug auf die Repräsentation der AWI-Forschungseinrichtungen und der Expeditionsfahrten im AWI-Intranet und den Internetseiten beschrieben.

Das AWI unterhält mehrere Forschungseinrichtungen. Dazu gehören unter anderem Forschungsschiffe, Forschungsstationen und Forschungsflugzeuge.

Unter der Rubrik „Schiffe und Stationen“ auf der AWI Homepage werden zu den einzelnen Forschungseinrichtungen Informationen angeboten. Man kann sich Einzelheiten über die vorhandenen Forschungsschiffe, Forschungsstationen, Flugzeuge, etc. ansehen.

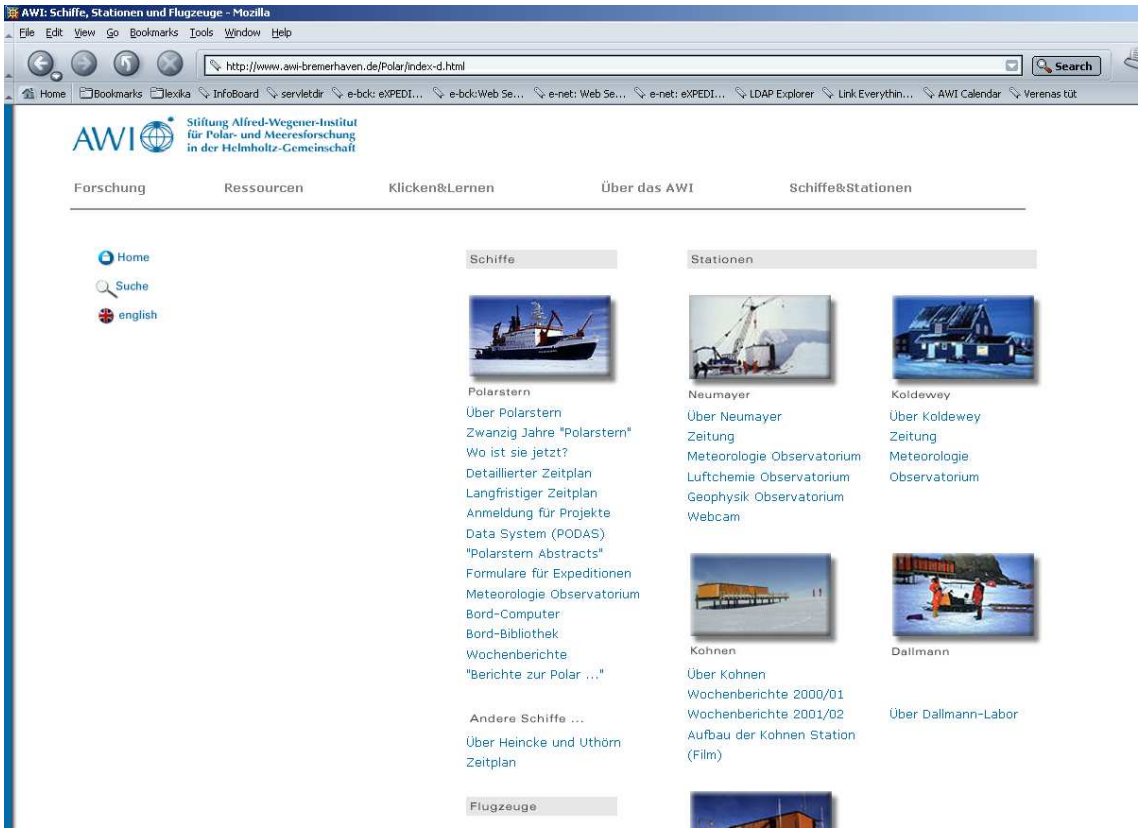

Abbildung 01: „Schiffe und Stationen“ auf der AWI Homepage

Zu den Expeditionsfahrten der einzelnen Forschungsschiffe gibt es für das Schiff „R.V. Polarstern“ einen Zeitplan. Dieser Zeitplan beinhaltet chronologisch eine Auflistung der

durchgeführten Expeditionsfahrten. Dabei gibt es einmal einen langfristigen Plan, der als PDF-Datei vorliegt, und einen dynamischen in PHP programmierten, der die Expeditionsfahrten für das einzelne Jahr auflistet. Die Auflistung erfolgt unter Angabe von Informationen über den Start-/Endhafen, Forschungsthema/-region und Dauer der Expedition.

Für jede Expedition existieren zusätzliche Informationen, die teilweise für den Benutzer versteckt sind bzw. dezentral auf der Homepage verteilt sind. Zu diesen Informationen gehören z.B. Wochenberichte, Fahrtzusammenfassungen, Pressemitteilungen, Karten mit dem Fahrverlauf und Publikationen.

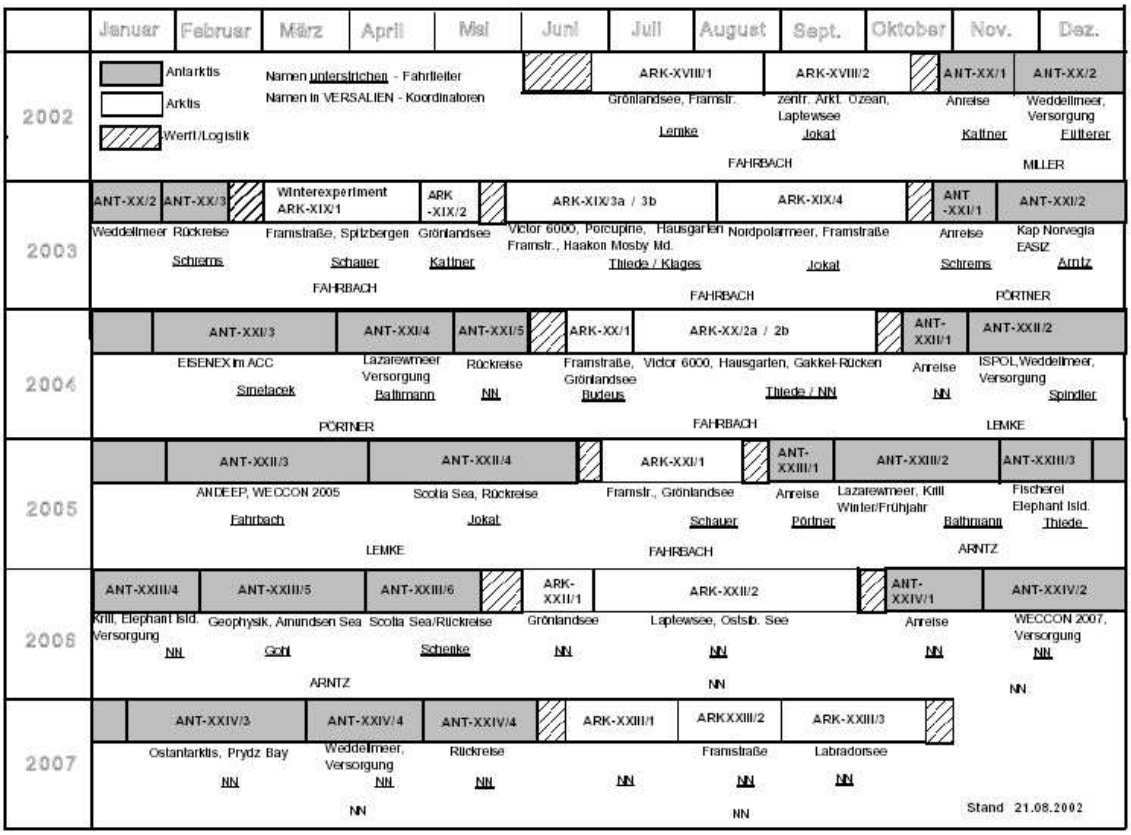

Abbildung 02: Langfristiger Zeitplan

Datengrundlage für den PHP Zeitplan sind die im Directory Server eingetragenen Expeditionsfahrten. Diese Daten werden manuell von den Administratoren in den Directory Server eingetragen.

3 Anforderungsbeschreibung

In der Anforderungsbeschreibung werden die Anforderungen der Auftraggeber an das zu entwickelnde System beschrieben. Dabei werden die quantitativen und qualitativen Produktanforderungen festgelegt.[3]

3.1 Funktionalität

Das zu entwickelnde System soll eine Repräsentation der am AWI durchgeführten Expeditionsfahrten und der eingesetzten Forschungseinrichtungen, bestehend aus den Forschungsschiffen, den Polarflugzeugen und den Forschungsstationen, bieten.

Das zukünftige System besteht aus einem öffentlichen und einem nichtöffentlichen Bereich. Im öffentlichen Bereich steht die Ausgabe und die Darstellung der Informationen zu den Expeditionsfahrten und den Forschungseinrichtungen im Vordergrund. Der nichtöffentliche Bereich ist der Administrationsbereich. Die Anwendung wird über das AWI interne Intranet sowie über die Internetseiten der AWI Homepage nutzbar sein. Der nichtöffentliche Bereich soll aus Sicherheitsgründen nur über die Intranetversion erreichbar sein.

Dabei werden im für alle Nutzer zugänglichen Bereich intern (Intranet) und extern (übers Internet) die gleichen Informationen angeboten.

3.1.1 Öffentlicher Bereich

Der öffentliche Bereich umfasst die Repräsentation der Expeditionsfahrten und der Forschungseinrichtungen. Dazu gehört die Ausgabe eines dynamischen Zeitplanes für die einzelnen Forschungsschiffe. Der Zeitplan soll alle Expeditionsfahrten des aktuell ausgewählten Schiffes chronologisch aufführen. Zu jeder aufgelisteten Expedition sollen Verweise auf die zusätzlichen Informationen aufgeführt werden.

Es können Informationen zu den verschiedenen Typen der Forschungseinrichtungen wie den Forschungsstationen oder Flugzeugen abgerufen werden. Ferner soll die Möglichkeit bestehen über eine Suchmaske den vorhandenen Datenbestand an eingetragenen Expeditionsfahrten anhand von bestimmten Kriterien zu durchsuchen.

Die Expeditionsfahrten können in einer Detailansicht angezeigt werden. Zu jeder Expedition können weiterführende Zusatzinformation angezeigt werden. Die

[3] vgl. [Balzert, 1999] S.8 ff

generierten Zeitpläne der Forschungsschiffe sollen als PDF-Datei exportiert werden können.

3.1.2 Nichtöffentlicher Bereich

Im nichtöffentlichen Administrationsbereich können die eingetragenen Expeditionsfahrten verwaltet werden. Die Expeditionsfahrten können gelöscht und modifiziert werden. Des weiteren können neue Expeditionsfahrten angelegt werden. Der Administrationsbereich soll nur für bestimmte Benutzergruppen zugänglich sein.

3.2 Benutzer

Die zukünftigen Benutzer lassen sich in zwei Gruppen einteilen. Eine Gruppe sind die Benutzer, die sich für die angebotenen Informationen interessieren oder sich über bestimmte Expeditionsfahrten informieren möchten. Diese Benutzer können institutsfremde Benutzer sein, die auf die Anwendung über das Internet über die AWI Homepage zugreifen. Es können aber auch Mitarbeiter des AWI sein, die die angebotenen Informationen abrufen wollen. Sie können die Anwendung auch über das AWI interne Intranet nutzen.

Die andere Benutzergruppe besteht aus den Mitarbeitern des AWI, die Beauftragte für die Expeditionsfahrten sind. Diese Benutzer nehmen eine Administratorrolle für den nichtöffentlichen Administrationsbereich ein. Sie sind berechtigt dort den Bestand an eingetragenen Expeditionsfahrten zu verwalten, d.h. sie dürfen z.B. Änderungen an den Expeditionsfahrten vornehmen, Einträge löschen oder neue Einträge anlegen.

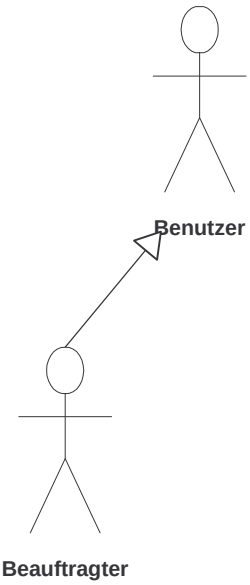

Abbildung 03: Benutzergruppen

3.3 Internationalisierung

Die Anwendung soll eine Sprachauswahl beinhalten. Es soll möglich sein die Anwendung in den Sprachen Deutsch und Englisch zu nutzen.

3.4 Wartbarkeit und Erweiterbarkeit

Das zu entwickelnde System soll besonders leicht zu warten, anzupassen und zu ergänzen sein. Der Quellcode soll verständlich, gut strukturiert und modular sein, um spätere Änderungen durch andere Programmierer zu erleichtern.

Der Anwendung sollen später leicht neue Komponenten, wie neue Forschungseinrichtungen, z.B. Forschungsschiffe, ohne größeren Programmieraufwand hinzugefügt werden können.

Des weiteren sollen die Textbezeichnungen in den Ausgaben nicht fest im Java-Code geschrieben sein, sondern durch den Einsatz von Konfigurationsdateien variabel bleiben.

3.5 Abgrenzung

Die Anwendung soll keine Plattform zum Austausch von Forschungsdaten darstellen. Es soll z.B. nicht möglich sein Daten über File Transfer Protocol (FTP) herunter- oder heraufzuladen. Zu diesem Zweck gibt es am AWI andere Dienste wie z.B. Datenbanken.

4 Was ist ein Portal?

Ein Portal ist eine Zusammenstellung von Internet-/Intranet- und/oder Ecommerce-Anwendungen, verschiedener Informationsquellen sowie anderer Anwendungen, die speziell auf eine Benutzergruppe zugeschnitten sind.

Ein Portal integriert verschiedene Dienste unter einheitlicher Oberfläche mit einheitlicher Bedienung. Die Navigation durch das Informationsangebot ist meist einfach gehalten. Viele Portale bieten dem Benutzer die Möglichkeit sich die Anwendung an eigene Bedürfnisse anzupassen, sie sich zu personalisieren. Durch Unterstützung von „Single sign on“, kann der Benutzer durch einmaliges Login auf alle Dienste zugreifen. Portale bieten oft eine Suchfunktion an, mit der das gesamte Angebot durchsucht werden kann. Die Repräsentation der Informationen erfolgt oft durch die dynamische Generierung von Webseiten. Die dazu eingesetzten Techniken sind beispielsweise Java Servlets und JSP sowie ASP oder PHP.

Portalseiten liefern dem Benutzer Informationen strukturiert und nach Kategorien geordnet.

Es lassen sich acht Funktionen ausmachen, die ein Portal charakterisieren. Diese Funktionen sind bei den heute existierenden Portalen mehr oder weniger stark ausgeprägt.

- Navigation, ein Portal soll den Benutzer bei seiner Suche nach Inhalten durch komfortable Such- und Navigationsmöglichkeiten unterstützen.
- Datenintegration, ein Portal soll über seine Oberfläche Zugriff auf Informationen aus den unterschiedlichsten Quellen gewähren.
- Personalisierung, jeder Benutzer kann sich den Portaleinstieg individuell an seine Bedürfnisse an die angebotenen Informationen anpassen.
- Notifikation, das Portal führt automatisch Notifikationen, Benachrichtigungen, durch, wenn bestimmte geschäftskritische Parameter über- oder unterschritten werden. Wenn z.B. bei der Durchführung eines Projektes bestimmte Fristen nicht eingehalten werden, kann das Portal automatisch die Verantwortlichen benachrichtigen.
- Wissensmanagement, das Portal soll gewährleisten, dass die Personen schnell und einfach Zugriff auf das für sie wichtige Wissen erhalten und sich nicht mühsam durch das gesamte Wissen arbeiten müssen.

- Workflow, die Workflow-Funktion beschreibt die Automatisierung von Geschäftsabläufen. Dies findet vor allem im Bereich E-Commerce Anwendung.
- Anwendungsintegration, verschiedene Anwendungen werden im Portal integriert. Oft bieten Portale ein Single-Sign-On, d.h. der Benutzer muss sich nur einmal am System anmelden und hat dann Zugriff auf alle Anwendungen für die er autorisiert ist.
- Infrastrukturdienste, Portale lassen sich erst durch die geeigneten Infrastrukturdienste, z.B. einen Webanwendungsserver, realisieren. Sie bilden die Basis für Portale. Das Portal soll jederzeit auf leistungsfähigere Systeme portiert werden können, ohne dass dazu grundlegende Veränderungen erforderlich wären.

4.1 Portalarten

Portale lassen sich nach den von ihnen angebotenen Informationen in folgende Gruppen einordnen.

4.1.1 Vertikale Portale

Vertikale Portale bieten Informationen zu einem speziellen Themenbereich an. Das Informationsangebot ist fokussiert, weiterführende Informationen werden auf dem Portal bereitgestellt.

4.1.2 Horizontale Portale

Horizontale Portale versuchen das gesamte Informationsangebot des Internets abzudecken. Ein wichtiges Kriterium ist hierbei die Personalisierung. Bei der Personalisierung werden die individuellen Wünsche des Benutzers berücksichtigt, indem er sich die Inhalte anpassen kann.[4]

4.2 Portal in Bezug auf diese Arbeit

Nach den genannten Kriterien zur Klassifizierung von Portalen, lässt sich das zu entwickelnde Expeditionsportal in die Gruppe der vertikalen Portale einordnen. Das Informationsangebot des Portals beschränkt sich auf den Themenbereich der am AWI durchgeführten Expeditionsfahrten und die Repräsentation der Forschungseinrichtungen des AWI. Es werden verschiedene Inhalte zum genannten Themenbereich integriert.

[4] vgl. [Portal, 2001]

Dazu gehören Angaben über Publikationen zu den einzelnen Expeditionsfahrten und den Forschungseinrichtungen sowie Angaben über beteiligte Personen oder Integration von meteorologischen Daten. Die eingebundenen Informationen stammen aus verschiedenen Datenquellen. Beispielsweise aus dem Directory Server, einer Datenbank oder aus Dateiverzeichnissen. Der Benutzer erhält durch Such- und Navigationsmöglichkeiten Hilfestellungen, um den Datenbestand an Informationen zu überblicken.

Der Aspekt der Personalisierung, nach dem sich der Benutzer das Informationsangebot und das Aussehen der Anwendung nach seinen Ansprüchen anpassen kann, wird hier nicht realisiert.

5 Einführung in verwendete Systeme

Nachfolgend werden die für die Erstellung der Anwendung verwendeten Systeme und Konzepte beschrieben.

5.1 Directory Server

Ein Verzeichnisserver ist eine spezialisierte Art von Datenbank. Ein Verzeichnisdienst ähnelt zwar in seiner Funktionsweise einer Datenbank, unterscheidet sich aber in einigen Punkten von traditionellen relationalen Datenbanken.

Ein Verzeichnis ist mehr auf das Finden und Auslesen von Informationen spezialisiert als auf das Schreiben immer neuer Informationen. Suchen ist eine der Hauptoperationen in Verzeichnisdiensten. Ein Verzeichnisdienst stellt daher fortgeschrittene Suchmöglichkeiten zur Verfügung. Verzeichnisse müssen in der Lage sein, eine große Anzahl von Suchanfragen zu verarbeiten; sie sind für den Lesezugriff optimiert. Schreibzugriff ist oft beschränkt auf die Administratoren oder Besitzer einer spezifischen Informationseinheit.

Daten in einem Directory Server werden in einem hierarchischem Baum gespeichert. Die Daten werden in Einträgen bestehend aus Attribut-Werte-Paaren abgelegt.

5.1.1 Directory Information Tree

Mit dem Begriff Directory Information Tree (DIT) wird der hierarchische Baum des Verzeichnisses bezeichnet. Er beinhaltet alle Einträge.

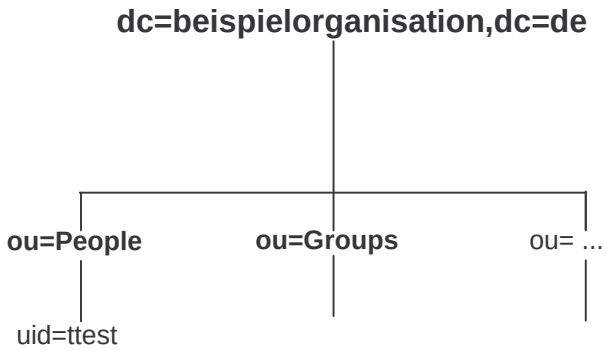

Abbildung 04: Beispiel für einen DIT

Jeder Eintrag im DIT lässt sich eindeutig über seinen Distinguished Name (DN) identifizieren und zuordnen. Der Eintrag im Zweig „People“ mit uid „ttest“ hat den DN dn=uid=ttest,ou=People,dc=beispielorganisation,dc=de.

5.1.2 Distinguished Name

Wie jede Datei in einem Dateisystem einen eindeutigen Pfad hat, so kann auch ein Eintrag in einem Verzeichnis eindeutig anhand seines Distinguished Names (DN) identifiziert werden. Der DN beschreibt den Eintrag durch bestimmte Attribute und ihre Werte. Der Pfad eines DNs wird im Vergleich zu der Pfadangabe einer Datei in einem Dateisystem in umgekehrter Reihenfolge dargestellt. In einem Dateisystem erfolgt die Darstellung des Pfades von links nach rechts, wobei die Wurzel links und der Dateiname ganz rechts steht. Der DN beschreibt den Pfad von rechts nach links, wobei das aktuelle Verzeichnisobjekt ganz links und die Wurzel ganz rechts steht.
Ein Beispiel für einen DN ist:

uid=kmindermann,ou=people,dc=organisationXY,dc=de

Dieser DN repräsentiert den Eintrag *uid=kmindermann* in dem Zweig *ou=people* in der Wurzel *dc=organisationXY,dc=de*. Der Wert, der in einem DN ganz links steht wird auch als Relative Distinguished Name (RDN) bezeichnet. Danach folgen die Attribute, die die Verzweigungsstelle über dem Eintrag kennzeichnen. Am Ende steht die Wurzel des Baumes.

5.1.3 Wurzeleintrag

Der Wurzeleintrag (Root Entry) ist der erste, bzw. oberste Eintrag in dem Verzeichnis.

5.1.4 Schema

Die Objektklassen eines Directory-Servers werden in einer zusammenfassenden Beschreibung, dem Schema, abgelegt. Ein Schema beschreibt welche Objektklassen in einem Verzeichnis erlaubt sind, welche Attribute sie haben müssen, welche sie haben dürfen und welche Syntax für die Attribute benutzt werden muss.
Jeder Eintrag muss eine Objektklasse haben. Die Objektklasse spezifiziert, welche Art von Objekt der Eintrag beschreibt und bestimmt welche Attribute er beinhaltet. Das Schema beschreibt die erforderlichen und erlaubten Arten von Attributen, ihre Struktur und Syntax. Das Standardschema kann verändert und erweitert werden, um es den eigenen Anforderungen anzupassen. So können z.B. neue Objektklassen angelegt werden.

5.1.4.1 Objektklassen und Attribute

Jeder Eintrag (Entry) in einem Directory beschreibt ein Objekt, also eine konkrete Instanz einer Objektklasse. Eine Objektklasse ist eine verallgemeinerte Beschreibung für dieses und gleichartige Objekte. Jeder Eintrag muss eine Objektklasse haben. Die Objektklasse spezifiziert, welche Art von Objekt der Eintrag beschreibt. Des weiteren verfügt die Objektklasse über eine Liste der zwingend vorgeschriebenen (required) und der nicht verbindlich vorgeschriebenen (allowed) Attribute.

Erforderliche Attribute sind die Attribute, die ein Eintrag enthalten muss, wenn die Objektklasse benutzt wird. Alle Einträge müssen das Attribut der Objektklasse beinhalten. Erlaubte Attribute sind die Attribute, die ein Eintrag haben kann, aber nicht zwingend haben muss.

Daten werden im Verzeichnisserver als Attribut-Werte-Paare repräsentiert. Jede Information, die im Verzeichnis gespeichert wird, ist mit einem Attribut verknüpft.[5]

5.1.5 Zugriffskontrolle

Die Zugriffskontrolle auf die Daten in einem Directory Server erfolgt über Access Control Information (ACI). Diese ACIs regeln die Zugriffsrechte auf das Directory. Dabei können Zugriffsrechte auf das gesamte Directory, einzelne Zweige oder auch auf einzelne Einträge gesetzt werden. Dabei wird angegeben für welche Benutzergruppen die Berechtigung gilt und welche Rechte gesetzt werden. Die Rechte die gesetzt werden können sind im einzelnen

- **Read,** gibt an, dass der Benutzer Leserechte hat.
- **Write,** gibt an, dass der Benutzer die Rechte hat einen Eintrag durch Hinzufügen, Ändern oder Löschen von Attributen zu modifizieren.
- **Add,** gibt an, dass der Benutzer das Recht hat neue Einträge anzulegen
- **Delete,** gibt an, dass der Benutzer das Recht hat Einträge zu löschen
- **Search,** der Benutzer hat das Recht Suchen im Directory durchzuführen. Der Benutzer muss Such- und Leserechte haben, um sich die Daten, die als Suchergebnis zurückgeliefert werden, ansehen zu können.
- **Compare,** gibt an, dass der Benutzer das Recht hat, Daten mit denen im Verzeichnis zu vergleichen.

[5] vgl. [HowSmiGor, 1999]

- **Selfwrite,** gibt an, dass der Benutzer das Recht hat seinen eigenen DN zu Gruppen hinzuzufügen oder zu entfernen.
- **All,** der angegebene DN hat alle Zugriffsrechte (read, write, search, delete, add, compare) für den angegeben Eintrag oder Zweig.

Die Rechte werden unabhängig voneinander erteilt. Ein Benutzer mit Schreibrechten (Add) kann zwar einen neuen Eintrag hinzufügen, ihn aber ohne die entsprechenden Löschrechte (Delete) nicht löschen.

Eine ACI sieht folgendermaßen aus

(target="ldap:///*Zweig im Directory*")(targetattr="*Attribute auf die diese ACI zutrifft*")(version 3.0; acl "*Name für die ACI*"; allow (welche Rechte werden gesetzt (add,...)) userdn = "ldap:///*userdn*";)

Über die folgenden Schlüsselwörter können Benutzer(-gruppen) für die ACI angegeben werden.

- **userdn= "ldap:///uid=xyz,ou=people,...",** der Benutzer kann durch einen DN angegeben werden
- **userdn = "ldap:///anyone"** - bestimmt „anonymous access“, anonymen Zugriff zugewähren bedeutet, dass jeder ohne sich am Directory Server anzumelden den Zugriff hat.
- **userdn = "ldap:///all"** - bestimmt „general access“, bedeutet, dass jeder der sich erfolgreich am Directory Server angemeldet hat, die angegebenen Rechte hat.
- **userdn = "ldap:///self"** - bestimmt „self access“, gibt an dass der Benutzer seinen eigenen Eintrag verändern darf.
- **userdn = "ldap:///parent"** - bestimmt „access for the parent entry“, gibt an, dass der Benutzer Zugriff auf den Eintrag hat, wenn sein DN der „parent“ des angegebenen Eintrages ist.

5.1.6 Authentifizierung

Es bestehen drei Methoden der Authentifizierung gegenüber dem Directory Server.

- Einfache passwortbasierte Authentifizierung (Simple, password-based authentication), in dieser Methode wird der Benutzername und das Passwort vom Client unverschlüsselt in Klartext an den Directory Server gesendet.

- Zertifikat-basierte Authentifizierung, in dieser Methode verbindet sich der Client mit dem Server über Secure Socket Layer (SSL) und der Client stellt ein Zertifikat zur Identifizierung.
- Authentifizierung unter Benutzung von Simple Authentication and Security Layer (SASL). SASL ist ein Schema, um eine Sicherheitsschicht für verbindungsbasierte Protokolle zu etablieren. Bei LDAP v3 ist dies die Verbindung zwischen dem Client und dem Directory Server. Dies setzt allerdings die Einbindung externer Authentifizierungsmethoden voraus.

Die zweite und dritte Methode benötigen eine Zertifizierungsstelle oder externe Authentifizierungsmechanismen. Die erste Methode ist sicher, wenn man mit SSL für eine Verschlüsselung sorgt oder wenn man sich in einem abgeschlossenen Netzwerk befindet.[6]

5.2 LDAP

Das Lightweight Directory Access Protocol (LDAP) wurde in den frühen 90ern an der University of Michigan als offener Standard für globale oder lokale Verzeichnisdienste im Netzwerk und im Internet entwickelt. LDAP ist ein TCP/IP-basiertes Directory-Zugangsprotokoll, das sich im Internet und in Intranets als Standardlösung für den Zugriff auf Netzwerk-Verzeichnisdienste für Datenbanken, E-Mails, Speicherbereiche und andere Ressourcen etabliert hat. LDAP ist von der Internet Engineering Task Force (IEFT) standardisiert und bietet einen einheitlichen Standard für Verzeichnisdienste (Directory Services).

Das LDAP-Protokoll definiert keinen Directory-Inhalt und auch nicht, wie der Directory Service erbracht werden soll. Es setzt direkt auf TCP/IP auf und arbeitet auf Client-Server-Basis. LDAP hat ein weltweit eindeutiges Format, in dem alle Namen darstellbar sind, es bietet unterschiedliche Layouts und eine eindeutige Zuordnung zwischen Namen und ihrer internen Repräsentation.

Zur Zeit sind LDAP-Verzeichnisdienste meistens in einzelnen großen Organisationen wie Hochschulen oder Unternehmen verbreitet.

[6] vgl. [WelDah, 2000]

5.3 Java

Java ist eine von Sun entwickelte objektorientierte Programmiersprache. Java ähnelt der Programmiersprache C++, verwendet eine ähnliche Syntax, verzichtet allerdings auf die prozessorspezifischen Anpassungen von C++. Gegenüber C++ ist Java portabler und leichter anwendbar, weil Java selber Speicher verwalten kann.
Sie ist plattformneutral und kann auf allen Rechnerplattformen eingesetzt werden, auf denen die Java Virtual Machine (JVM) implementiert ist.

5.3.1 Servlets

Die Firma Sun stellt mit der Servlet API eine Methode bereit, um serverseitige Javaprogramme zu erstellen. Servlets sind serverseitige Javaprogramme, die die Funktionalität des Webservers erweitern. Mit Servlets ist es möglich dynamische Webseiten zu erzeugen. Über ein Servlet kann ähnlich wie über CGI auf die Umgebungsvariablen zugegriffen werden. Servlets haben den Vorteil, dass sie unabhängig von der Serverumgebung einsetzbar sind, d.h. sie können ohne Probleme auf andere Webserver portiert werden. Außerdem bieten Servlets dem Programmierer alle Möglichkeiten, die die Programmiersprache Java bereitstellt. Des weiteren unterstützt die Servlet API Cookies und Sessionmanagement, um Daten über mehrere Anfragen zu speichern.
Um Servlets zu nutzen wird ein Webserver benötigt, der die Servlet API unterstützt. Ein Servlet kommuniziert mit einem Client, z.B. ein Webbrowser, über Anfrage- und Antwort-Objekte (Request und Response), die im Servlet-Container implementiert sind. Diese Anfrage- und Antwort-Objekte können sowohl protokollunabhängig sein, als auch auf dem HTTP-Protokoll basieren.
Beispielsweise stellt ein Client eine Anfrage an das Servlet, worauf das Servlet eine dynamische Seite als Antwort generiert und an den Browser sendet. Das Servlet kann dabei den dynamischen Inhalt auch aus einer Datenbank erhalten. Die folgende Abbildung zeigt diesen Ablauf zwischen Client, Webserver mit dem Servlet und Datenbank auf.

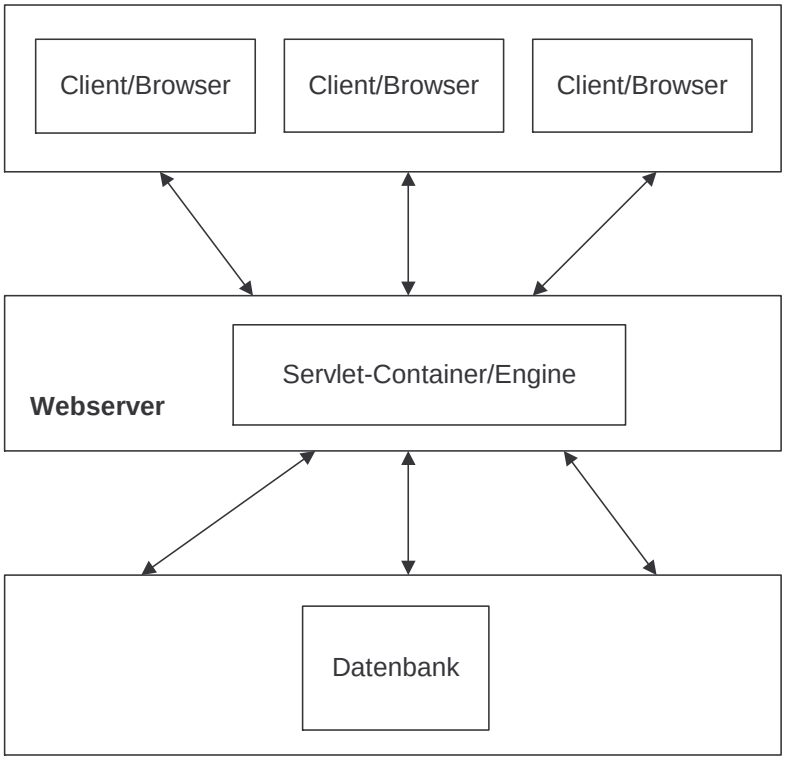

Abbildung 05: Ablauf zwischen Client-Server-Datenbank (Servlet)

Der Servlet-Container, bzw. die Servlet-Engine, ist ein Teil des Webservers, der die Netzwerkdienste zum Empfangen von Anfragen und Senden von Antworten bereitstellt und die Servlets über ihren gesamten Lebenszyklus enthält und verwaltet.

Alle Klassen, die die Schnittstelle *javax.servlet.Servlet* implementieren, haben einen festen Lebenszyklus. Er beschreibt, wie ein Servlet instanziiert und initialisiert wird, wie es Anfragen von Clients entgegennimmt und wie das Servlet beendet wird.

Der Servlet-Container übernimmt die Instantiierung eines Servlets. Das Servlet wird dabei in einen Zustand versetzt, in dem es bereit ist, Anfragen von einem Client z.B. einem Webbrowser zu beantworten.

Nach der Instantiierung des Servlets folgt die Initialisierung durch den Servlet-Container. Dies erfolgt durch den Aufruf der Methode *init()*. Durch Überschreibung dieser Methode ist es möglich, Operationen durchzuführen, die nur einmal vom Servlet durchgeführt werden sollen. Beispielsweise das Herstellen von Datenbankverbindungen oder der Zugriff auf Konfigurationsdaten.

Nach erfolgreicher Initialisierung ist das Servlet bereit, Anfragen zu beantworten. Dies geschieht durch Aufruf der *service()*-Methode. Je nach Art der Anfrage, wird von der *service()*-Methode die entsprechende *doGet()*- bzw. *doPost()*-Methode aufgerufen, um

auf die Anfrage zu antworten. Für jede Anfrage eines Clients wird ein neuer Thread erzeugt. Die *service()*-Methode kann beliebig oft aufgerufen werden.

Das Servlet wird vom Servlet-Container durch Aufruf der *destroy()*-Methode beendet. Durch Überschreiben dieser Methode können z.B. noch geöffnete Datenbankverbindungen geschlossen werden. Nach Aufruf der Methode *destroy(),* kann keine Anfrage mehr an das Servlet gestellt werden. Die Methode *destroy()* wird aufgerufen, wenn eine gewisse Zeit lang keine Anfragen mehr an das Servlet gestellt wurden oder wenn der Webserver gestoppt wird.

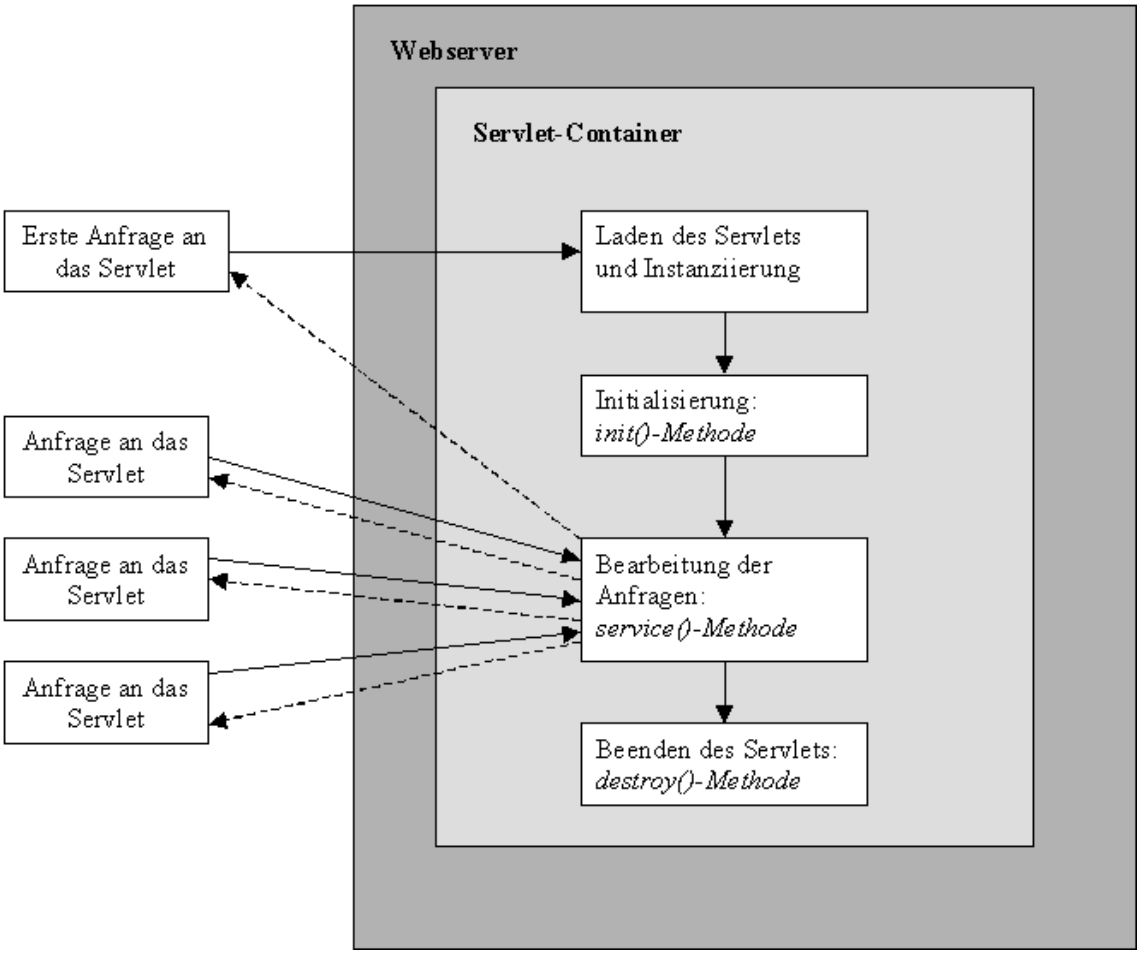

Abbildung 06: Lebenszyklus eines Servlets

5.4 HTML

Dokumente innerhalb des World Wide Webs (www) werden in der Dokument-Beschreibungssprache Hypertext Markup Language (HTML) definiert. Diese Sprache beschreibt den Aufbau eines Dokuments mit Absätzen, Überschriften, Verweisen zu anderen Dokumenten, etc. HTML-Dokumente enthalten nur Text und lassen sich

Rechner- und plattformunabhängig bearbeiten und anzeigen. Zum Anzeigen von HTML-Dokumenten wird ein Browser benötigt.

5.5 Javascript

JavaScript ist eine Scriptsprache, die von Netscape eingeführt wurde. JavaScript dient als Schnittstelle zwischen dem Benutzer und der Anwendung. Unter anderem wird sie benutzt, um optisch ansprechende Interaktionen (z.B. in den Navigationselementen) einzubringen und so das Design einer Homepage anspruchsvoller zu gestalten.

6 Arbeits- und Entwicklungsumgebung

Nachfolgend wird die Infrastruktur der Arbeits- und Entwicklungsumgebung am AWI beschrieben. Dabei werden insbesondere die Hardwareumgebung und die Softwareumgebung, die für die Entwicklung der Anwendung von Relevanz sind, betrachtet. Andere Hardware- und Softwarekonfigurationen am AWI werden nicht weiter beschrieben, da sie für diese Arbeit nicht direkt von Bedeutung sind.

6.1 Hardwareumgebung

Am AWI werden mehrere Server eingesetzt. Für dieses Arbeit ist besonders der „Enterprise 10000“ relevant. Auf diesem Server sind der Directory Server und mehrere Webserver installiert. Des weiteren befinden sich auf ihm Dateiverzeichnisse mit für die Anwendung wichtigen Daten.

Die Domäne e-net auf dem „Enterprise 10000“, auf der die Anwendung laufen wird, ist mit 4 Prozessoren a 333MHz und mit 4GB RAM ausgestattet.

Am AWI werden Arbeitsplatzrechner mit den folgenden Betriebssystemen eingesetzt

- Solaris 2.8 und 2.9
- Windows NT/2000/XP
- MacOS 9, X.

6.2 Softwareumgebung

6.2.1 Webserver

Am AWI wird ein Webserver der Firma iPlanet in der Version 4.1_sp7 zur Präsentation der Intranet-/Internet-Inhalte eingesetzt. Dieser Webserver gestattet die Nutzung von Schnittstellen wie das Common Gateway Interface (CGI), Java Servlets und JSP.

6.2.2 Java

Auf dem Webserver ist das Java Development Kit (JDK) in der Version 1.2 und einige zusätzliche Bibliotheken wie die Servlet API, PDFLib und das LDAPSDK installiert. Zum Erstellen des Java-Codes wird der JBuilder 8 in der Personal Edition verwendet.

6.3 Client-Konfigurationen

Es sind folgende Client-Rechnerkonfigurationen der Benutzer am AWI möglich.

- PC (WIN NT/2000/XP)
- Mac (Mac OS 9, 10)
- Sun (Solaris 2.8, Solaris 2.9)

mit den Browsern

- Netscape Navigator
- Internet Explorer
- Mozilla

Da die Anwendung auch über das Internet nutzbar sein wird, können alle denkbaren Client-Konfigurationen möglich sein.

6.4 Intranet- und Internetrepräsentation des AWI

6.4.1 Intranet

Im Intranet werden hauptsächlich Informationen für die Mitarbeiter des AWI bereitgestellt. Dazu gehören „Schwarze Bretter“, Veranstaltungskalender, Publikationsdatenbank, Telefonlisten, persönliche Homepages sowie ein Programm zum Verwalten der Mitarbeiterinformationen. Der Großteil dieser Anwendungen wurden in den Programmiersprachen Perl, PHP und Java erstellt. Sie generieren dynamisch die Ausgaben und beziehen ihre Daten aus dem Directory Server.

6.4.2 Internet

Über die Internetpräsenz präsentiert sich das AWI nach außen. Auf den Internetseiten werden allgemeine Informationen über das AWI bereitgestellt. Interessierte Benutzer können sich hier über die Tätigkeiten und die Forschungsbereiche des AWI informieren.

6.5 Directory Server

Am AWI wird ein Directory Server der Firma iPlanet in der Version 5.1 eingesetzt.
Der Verzeichnisserver wird am AWI zur Speicherung von verschiedenen Informationen und Daten eingesetzt. Beispielsweise werden in ihm Daten zur Mitarbeiterverwaltung, Informationen über Publikationen oder über die verschiedenen Fachbereiche gespeichert.

6.5.1 Struktur und Schema

Die gespeicherten Informationen werden abgehend von dem Wurzeleintrag in verschiedenen Zweigen im Directory Server abgelegt. Die folgende Abbildung zeigt die grobe Struktur des Verzeichnisbaumes.

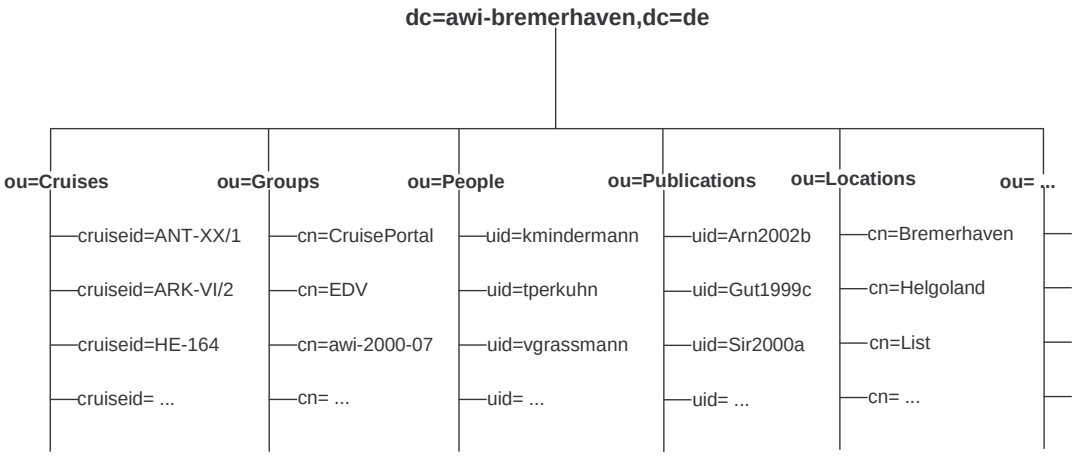

Abbildung 07: Beispielhafte Verzeichnisstruktur des Directory Servers am AWI

Für die zu erstellende Anwendung sind nur einige der Zweige des Directory Servers relevant. Dies sind im einzelnen

- ou=Cruises, in diesem Zweig werden die Expeditionsfahrten der Forschungsschiffe abgelegt.
- ou=Groups, in diesem Zweig werden Gruppen definiert, z.B. sind hier die Fachbereiche und ihre Unterbereiche aufgeführt. Daneben werden Personen bestimmten Gruppen zugeordnet.
- ou=People, hier werden Informationen über die Mitarbeiter in Personeneinträgen abgelegt. Dazu gehören z.B. Vorname, Nachname, Fachbereich, Raum und Telefonnummer.
- und ou=Publications, in diesem Zweig werden Informationen über die am AWI erstellten Publikationen in Publikationseinträgen abgelegt.

6.5.2 Objektklassen

Für die Anwendung sind einige Objektklassen des Directory Server Schemas wichtig. Das sind zum Teil Objektklassen des Standardschemas und vom AWI selbst definierte Objektklassen.

6.5.3 Expeditionseintrag

Im folgenden wird beschrieben wie ein Expeditionseintrag im Directory Server aussieht. Ein Expeditionseintrag wird innerhalb des Zweiges eindeutig durch seine cruiseID gekennzeichnet. Grundlage für einen Eintrag sind die Objektklassen „top“ und „AWICruise“. Eine Objektklasse definiert, um welche Art von Eintrag es sich handelt.

6.5.3.1 Objektklasse „top“

Die Objektklasse „top“ ist die Basis für alle anderen Objektklassen. Alle anderen Objektklassen erben von ihr.

```
objectclass top
        oid 2.5.6.0
        requires
                objectClass
        allows
                aci
```

6.5.3.2 Objektklasse „AWICruise“

Die Objektklasse „AWICruise“ beschreibt die Instanz eines Expeditionseintrages. Sie definiert welche Attribute ein Expeditionseintrag zwingend haben muss und welche Attribute er haben kann.
Zwingend erforderliche Attribute der Objektklasse sind die Attribute “objectclass” und “CruiseID”. Alle anderen aufgeführten Attribute sind optional.

```
objectclass AWICruise
        oid AWICruise-oid
        superior top
        requires
                objectclass,
                Cruiseid
        allows
                CruiseStatus,
                CruiseShip,
                CruiseBeginn,
                CruiseEnd,
```

CruisePortBeginn,
CruisePortEnd,
CruisePortStop,
CruiseRegion,
CruisePressRelease,
CruiseNews,
CruiseResearch,
CruiseResearchSummary,
CruiseCoordinator,
CruisePI,
CruiseCaptain,
CruiseHeli

Abbildung 08: Definition der Objektklasse „AWICruise“

Die folgende Tabelle beschreibt die Attribute, die die Objektklasse „AWICruise“ umfasst.

Attributname	**Beschreibung**	**Beispiel**	**Typ**	**mehr-wertig**	**Kontrolliertes Vokabular**
CruiseID	Eindeutig identifizierbarer Name innerhalb des Zweiges	ANT-XX/1	cis	nein	- für Polarstern: ANT-,ARK-<Expnr. in römischen Zahlen>/<Fahrt-abschnitt in natürlichen Zahlen> - für Heincke: HE-<Expnr.in natürlichen Zahlen> - Wartung: P-/HE-<Wartungs-beginn in Form jjjj.mm.tt>
CruiseShip	Name des Forschungsschiffes	Polarstern	cis	nein	- Polarstern - Heincke
CruiseStatus	Status des Expeditionseintrages	public	cis	nein	- public - nonPublic
CruiseBeginn	Startdatum der Expedition	24.02.2001	cis	nein	in Form: tt.mm.jjjj
CruiseEnd	Enddatum der Expedition	12.04.2001	cis	nein	in Form: tt.mm.jjjj
CruisePortBeginn	Abfahrtshafen	Punta Arenas	cis	nein	nein
CruisePortEnd	Ankunftshafen	Tromsø	cis	nein	nein
CruisePortStop	Zwischenstophafen	Neumayer	cis	nein	nein
CruiseRegion	Forschungsregion	Arctic Ocean	cis	nein	nein
CruiseResearch	Forschungsbereich	Physical Oceanography	cis	nein	nein

Attributname	Beschreibung	Beispiel	Typ	mehr-wertig	Kontrolliertes Vokabular
CruiseResearch Summary	Kurzfassung	http://.../ark-XIX-1.html	cis	nein	HttpLink
CruisePressRelease	Pressemitteilung	http://.../pressrelease.html	cis	ja	HttpLink
CruiseNews	Wochenberichte	http://.../report1.html	cis	ja	HttpLink
CruiseCoordinator	Koordinator	hmiller	cis	nein	uid oder Nachname
CruisePI	Fahrtleiter	rgersonde	cis	nein	uid oder Nachname
CruiseCaptain	Kapitän	Pahl	cis	nein	uid oder Nachname
CruiseHeli	Anzahl der mitgeführten Helikopter	CruiseHeli=2	cis	nein	0-2

Tabelle 01: Attribute der Objektklasse „AWICruise“

6.5.4 Personeneintrag

Die Einträge über die AWI-Mitarbeiter werden in dem Zweig ou=People im Directory Server gespeichert. Ein Personeneintrag setzt sich aus verschiedenen Objektklassen zusammen. Dies sind im einzelnen top, person, organizationalPerson, eduPerson, inetOrgPerson, awiPerson, mailRecipient, nsLicense, nsMessagingServerUser. Es wird hier darauf verzichtet, alle möglichen Attribute eines Personeneintrages aufzulisten. Im folgenden werden nur die unmittelbar für die Anwendung wichtigen Attribute aufgeführt.

Relevante Attribute eines Personeneintrages sind

- uid, Unique Identifier, ist der eindeutige Bezeichner innerhalb des Zweiges. z.B. uid=kmindermann
- givenname, der Vorname der Person
- sn, Abkürzung für surname, der Nachname der Person
- userPassword, speichert das Passwort der Person

6.5.5 Publikationseintrag

Die Einträge über die AWI-Publikationen werden in dem Zweig ou=Publications abgelegt. Ein Publikationseintrag setzt sich aus den Objektklassen top, person, organizationalPerson und PublicationClass zusammen.

Nachfolgend werden die relevanten Attribute für die Expeditionsanwendung beschrieben.

Relevante Attribute eines Publikationseintrages sind

- cruisedn, beinhaltet die DNs der Expeditionseinträge, die mit dieser Publikation zu tun haben, z.B. cruiseid=ANT-XX/2,ou=Cruises,dc=awi-bremerhaven,dc=de
- uid, Unique Identifier, ist der eindeutige Bezeichner innerhalb des Zweiges. z.B. uid=Ada2001a
- publicationauthor, hat als Wert die Autoren der Publikation
- publicationtype, beschreibt den Typ der Publikation, z.B. book oder article
- publicationyear, beinhaltet das Erscheinungsjahr der Publikation

6.6 Bestehende Javaanwendungen am AWI

Am AWI wurden im Rahmen von vorherigen Diplomarbeiten webbasierte Java-Anwendungen entwickelt. Die bereits bestehenden Anwendungen bilden ein Rahmenwerk, an dem man sich orientieren kann. Da das zu entwickelnde System Berührungspunkte mit beiden Anwendungen hat und gemeinsame Ressourcen genutzt werden, werden diese im folgenden kurz beschrieben.

6.6.1 ePIC

Bei dieser Anwendung handelt es sich um ein System, dessen Gegenstand die Verwaltung von elektronischen Publikationen ist. Dies umfasst im einzelnen die Suche nach Publikationen, das Löschen, Neueintragen sowie das Modifizieren von Publikationen. Die Daten über die Publikationen sind in Publikationseinträgen im Directory Server gespeichert und werden für die Repräsentation und Verarbeitung entsprechend aufbereitet.

6.6.2 ePersonal

ePersonal ist eine Anwendung mit der die Mitarbeiter des AWI ihre Personendaten ändern können. Die Daten der AWI-Mitarbeiter werden im Directory Server erfasst. Jeder Mitarbeiter hat einen Personeneintrag, in dem Informationen über den Fachbereich in dem er arbeitet, der Ort, Raum, Telefonnummer oder Emailadresse gespeichert sind. Der Mitarbeiter hat die Möglichkeit über die Anwendung die Daten in seinem Personeneintrag zu ändern. Diese Daten sind unter anderem Grundlage für die dynamisch generierte Telefonliste. Die Anwendung erzeugt außerdem dynamisch für jeden Mitarbeiter eine „Personal Homepage“, persönliche Homepage, die unter Angabe des Namens, Titel, Telefonnummer, etc. aufführt wo und in welchem Fachbereich er arbeitet.

7 Objektorientierte Analyse

Das Ziel der Analyse ist es, Wünsche und Anforderungen an das neue System zu ermitteln und zu beschreiben.

Die OOA dient der Ermittlung und Beschreibung der Anforderungen an ein Software-System mittels objektorientierter Konzepte und Notationen. Das Ergebnis ist ein OOA-Modell. Eine detaillierte verbale Beschreibung der gewünschten Anforderungen an das zu entwickelnde Expeditionsportal wurde bereits in Kapitel 3 vorweggenommen. [7]

7.1 Pflichtenheft

Das Pflichtenheft enthält eine Zusammenfassung aller fachlichen Anforderungen, die das zu entwickelnde Software-Produkt aus Sicht des Auftraggebers erfüllen muss. Es ist das Ergebnisdokument einer Anforderungsdefinition.[8]

Anhand der bereits beschriebenen Anforderungen wurde das folgende Pflichtenheft erstellt.

7.1.1 Zielbestimmung

7.1.1.1 Muss-Kriterien

- Ausgabe eines dynamischen Schiffzeitplanes
- Suche nach Expeditionsfahrten
- Detailansicht von Expeditionseinträgen
- Ausgabe von Informationen zu den einzelnen Forschungseinrichtungen
- Administrationsfunktion
  - Neueintragung von Expeditionseinträgen
  - Modifikation von Expeditionseinträgen
  - Löschen von Expeditionseinträgen
- Exportierung des Zeitplanes als PDF-Datei
- Internationalisierung
- dynamische Anwendungskomponenten

[7] vgl. [Balzert, 1999] Seite 8 ff.
[8] vgl. [HelBalzert, 1996] Seite 104 ff.

7.1.1.2 Kann-Kriterien

- Benutzerhandbuch

7.1.1.3 Abgrenzungskriterien

- keine

7.1.2 Einsatz

7.1.2.1 Anwendungsbereiche

- Intranet
- Internet

7.1.2.2 Zielgruppen

- Interessierte Benutzer
- Beauftragte

7.1.2.3 Betriebsbedingungen

- tägliche Nutzung

7.1.3 Umgebung

7.1.3.1 Software

- Betriebssystem: SUN Solaris 8.0
- iPlanet Webserver 4.1 sp_7
- iPlanet Directory Server 5.1
- jdk 1.2

7.1.3.2 Hardware

- Server Plattform: SUN Enterprise 10000
- Client Plattform: Kann nicht eindeutig angegeben werden

7.1.3.3 Orgware

- LDAP-Schema
- Daten im Directory Server

7.1.4 Funktionalität

Die Funktionalität umfasst die Repräsentation der im Directory Server gespeicherten Daten der Expeditionsfahrten der AWI-Forschungsschiffe und der AWI-Forschungseinrichtungen wie z.B. Forschungsschiffe und Forschungsstationen, sowie einen Administrationsbereich zur Verwaltung der Einträge der Expeditionsfahrten.

7.1.5 Daten

Datengrundlage sind die im Directory Server erfassten Expeditionsfahrten der AWI Forschungsschiffe seit 1998.

7.1.6 Leistungen

Schnelle Ausgabe der Ergebnisse ohne Verzögerungen

7.1.7 Benutzeroberfläche

Die Benutzeroberfläche wird als Webanwendung auf einem Browser dargestellt. Die Bedienung soll eindeutig und selbsterklärend sein.

7.1.8 Qualitätsziele

- Hohe Änderbarkeit
- Ausgabe auf verschiedenen Plattformen
- Benutzungsfreundlichkeit

7.1.9 Ergänzungen

- keine

7.2 OOA-Modell

Das OOA-Modell bildet die fachliche Lösung des zu realisierenden Systems, die in einer objektorientierten Notation modelliert wird. Das OOA-Modell bildet das wichtigste Ergebnis der Analyse.

7.2.1 Expeditionssystem

Nach der Analyse lassen sich fünf Geschäftsprozesse identifizieren. Des weiteren kann ein Paket gebildet werden, das die Geschäftsprozesse umfasst, die die Verwaltungsfunktionen des Systems beschreiben.

Das Expeditionssystem umfasst das Paket Expeditionsverwaltung und fünf Geschäftsprozesse.

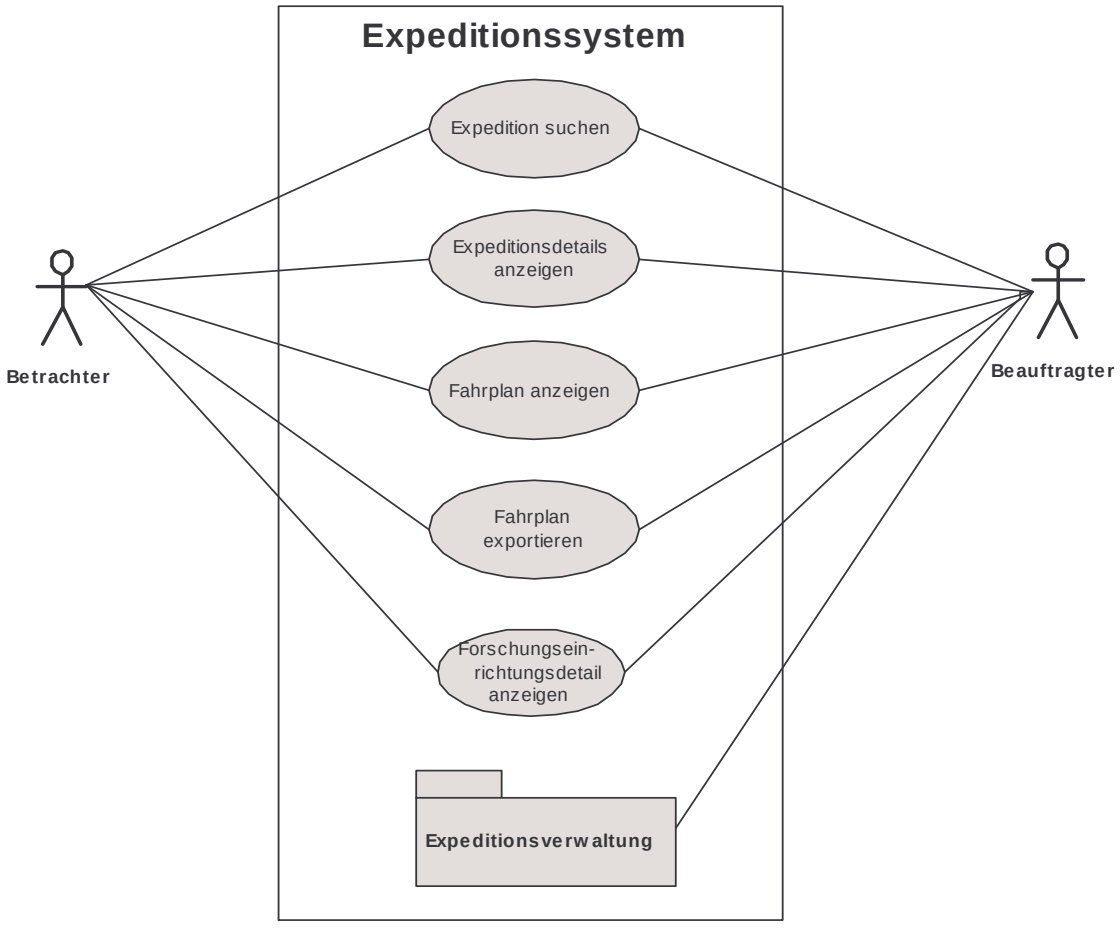

Abbildung 09: Geschäftsprozessdiagramm Expeditionssystem

7.2.2 Beschreibung der Geschäftsprozesse

Nachfolgend werden die identifizierten Geschäftsprozesse beschrieben.

7.2.2.1 Geschäftsprozess „Expedition suchen“

Geschäftsprozess:	Expedition suchen
Ziel:	Bereitstellung von Suchdaten
Kategorie:	primär
Vorbedingung:	vorhandene Datensätze
Nachbedingung Erfolg:	Auflistung des Suchergebnisses
Nachbedingung Fehlschlag:	kein passender Eintrag
Akteure:	Betrachter, Beauftragter
Auslösendes Ereignis:	Benutzer stellt Suchanfrage
Beschreibung:	1 Eingabe der Suchkriterien 2 Erstellung des Suchfilters 3 Durchführung der Suche 4 Ausgabe des Suchergebnisses
Erweiterungen:	4a Detailansicht
Alternativen:	1a Abfangen ungültiger Eingaben

Tabelle 02: Beschreibung des Geschäftsprozesses „Expedition suchen“

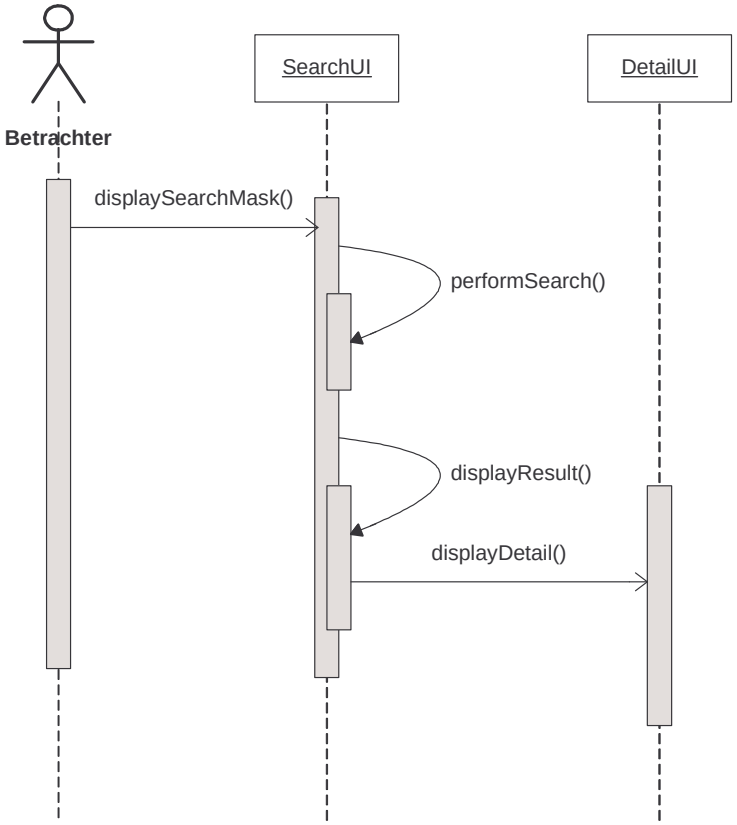

Abbildung 10: Sequenzdiagramm „Expedition suchen“

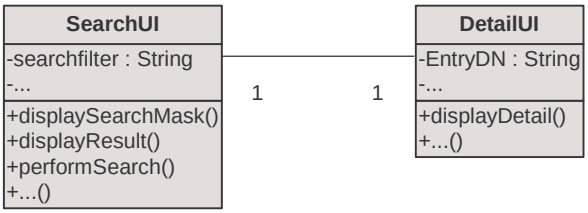

Abbildung 11: Klassendiagramm „Expedition suchen“

7.2.2.2 Geschäftsprozess „Expeditionsdetails anzeigen“

Geschäftsprozess:	Expeditionsdetails anzeigen
Ziel:	Bereitstellung von Expeditionsdaten
Kategorie:	primär
Vorbedingung:	vorhandene Datensätze
Nachbedingung Erfolg:	Anzeige der Expeditionsdetails
Nachbedingung Fehlschlag:	-
Akteure:	Betrachter, Beauftragter

Auslösendes Ereignis:	Benutzer will detaillierte Informationen zu einer Expedition ansehen
Beschreibung:	1 Expeditionsfahrt auswählen 2 Ausgabe der detaillierten Informationen zu der gewählten Expedition
Erweiterungen:	-
Alternativen:	-

Tabelle 03: Beschreibung des Geschäftsprozesses „Expeditionsdetails anzeigen“

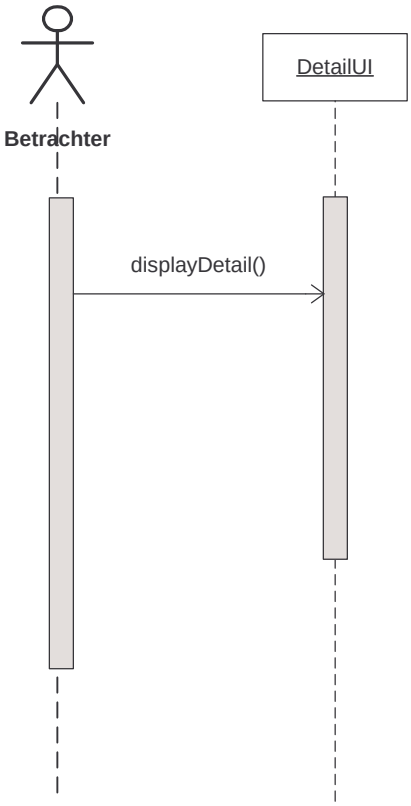

Abbildung 12: Sequenzdiagramm „Expeditionsdetails anzeigen“

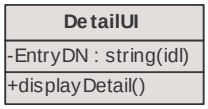

Abbildung 13: Klassendiagramm „Expeditionsdetails anzeigen“

7.2.2.3 Geschäftsprozess „Zeitplan anzeigen“

Geschäftsprozess:	Zeitplan anzeigen

Ziel:	Chronologische Ausgabe der Expeditionsfahrten
Kategorie:	primär
Vorbedingung:	vorhandene Datensätze
Nachbedingung Erfolg:	anzeigen aller Expeditionsfahrten für das gewählte Schiff und Jahr
Nachbedingung Fehlschlag:	keine Daten für das Schiff und Jahr vorhanden
Akteure:	Betrachter, Beauftragter
Auslösendes Ereignis:	Benutzer will sich den Schiffsplan für ein bestimmtes Schiff und Jahr ansehen
Beschreibung:	1 Auswahl des Expeditionsschiffes und des Expeditionsjahres 2 Erstellung des Suchfilters 3 Durchführung der Suche 4 Ausgabe des Zeitplanes
Erweiterungen:	4a Detailansicht der Expeditionen
Alternativen:	-

Tabelle 04: Beschreibung des Geschäftsprozesses „Zeitplan anzeigen“

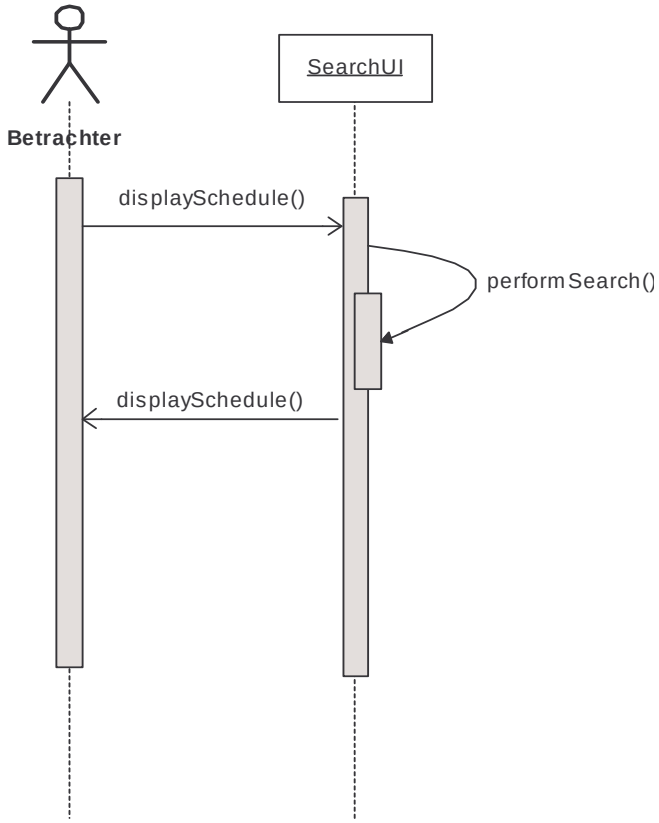

Abbildung 14: Sequenzdiagramm „Zeitplan anzeigen“

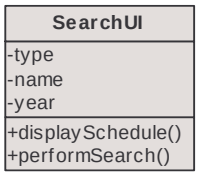

Abbildung 15: Klassendiagramm „Zeitplan anzeigen“

7.2.2.4 Geschäftsprozess „Zeitplan exportieren“

Geschäftsprozess:	Zeitplan exportieren
Ziel:	Zeitplan des gewählten Schiffes und Jahres als PDF-Datei exportieren
Kategorie:	primär
Vorbedingung:	vorhandene Datensätze
Nachbedingung Erfolg:	generierte PDF-Datei
Nachbedingung Fehlschlag:	Generierung einer PDF-Datei

	fehlgeschlagen
Akteure:	Betrachter, Beauftragter
Auslösendes Ereignis:	Benutzer möchte eine PDF-Datei mit dem Zeitplan des gewählten Schiffes und Jahres
Beschreibung:	1 Auswahl des Expeditionsschiffes und des Expeditionsjahres 2 Erstellung des Suchfilters 3 Durchführung der Suche 4 Generierung der PDF-Datei
Erweiterungen:	-
Alternativen:	-

Tabelle 05: Beschreibung des Geschäftsprozesses „Zeitplan exportieren“

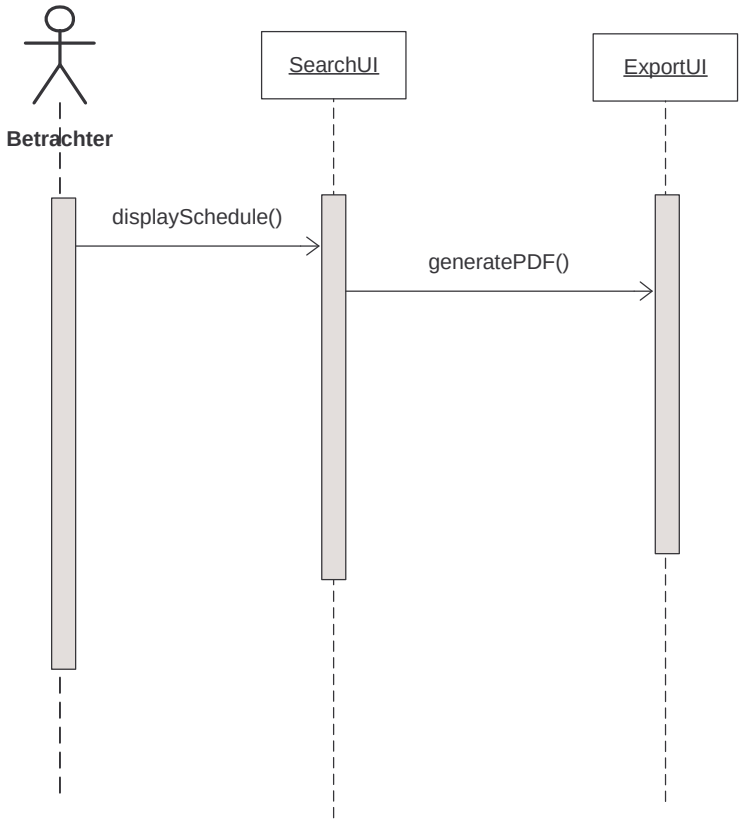

Abbildung 16: Sequenzdiagramm „Zeitplan exportieren“

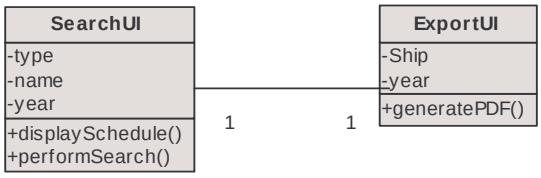

Abbildung 17: Klassendiagramm „Zeitplan exportieren“

7.2.2.5 Geschäftsprozess „Forschungseinrichtungsdetails anzeigen“

Geschäftsprozess:	Forschungseinrichtungsdetails anzeigen
Ziel:	Ausgabe von Informationen zu der ausgewählten Forschungseinrichtung
Kategorie:	primär
Vorbedingung:	vorhandene Datensätze
Nachbedingung Erfolg:	Informationen zu der gewählten Forschungseinrichtung werden angezeigt
Nachbedingung Fehlschlag:	-
Akteure:	Betrachter, Beauftragter
Auslösendes Ereignis:	Benutzer will sich Informationen über eine bestimmte Forschungseinrichtung, z.B. über ein bestimmtes Schiff, ansehen
Beschreibung:	1 Auswahl der Forschungseinrichtung 2 Erstellung des Suchfilters 3 Durchführung der Suche 4 Ausgabe der spezifischen Informationen
Erweiterungen:	-
Alternativen:	-

Tabelle 06: Beschreibung des Geschäftsprozesses „Forschungseinrichtungsdetails anzeigen“

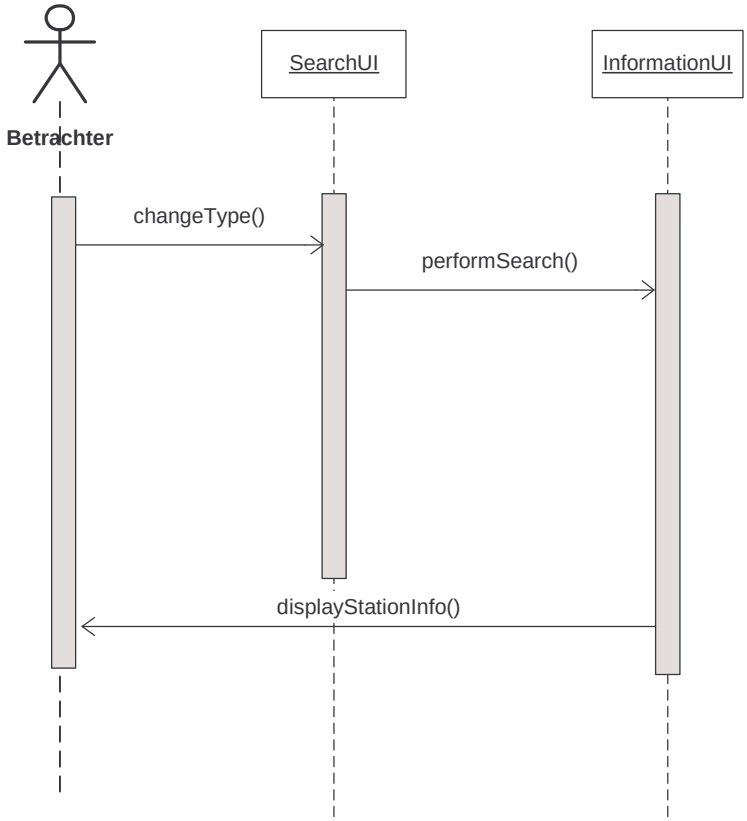

Abbildung 18: Sequenzdiagramm „Forschungseinrichtungsdetails anzeigen“

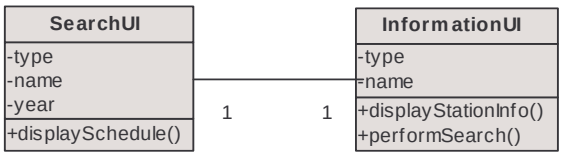

Abbildung 19: Klassendiagramm „Forschungseinrichtungsdetails anzeigen“

7.2.3 Paket Expeditionsverwaltung

Das Paket „Expeditionsverwaltung“ umfasst die Geschäftsprozesse, die die Verwaltungsfunktionen des Systems beschreiben.

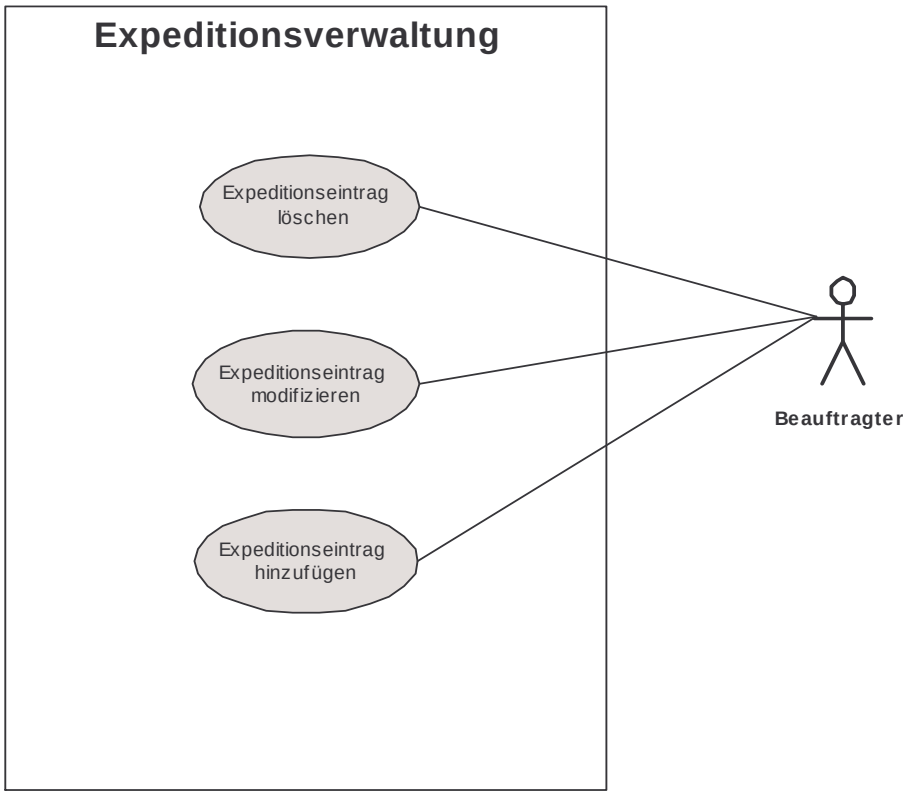

Abbildung 20: Geschäftsprozessdiagramm Paket Expeditionsverwaltung

7.2.4 Beschreibung der Geschäftsprozesse

7.2.4.1 Geschäftsprozess „Expeditionseintrag löschen“

Geschäftsprozess:	Expeditionseintrag löschen
Ziel:	Verwaltung der Expeditionsdaten, Expeditionseintrag löschen
Kategorie:	primär
Vorbedingung:	zu löschender Eintrag muss existieren
Nachbedingung Erfolg:	Eintrag wurde gelöscht
Nachbedingung Fehlschlag:	Eintrag existiert nicht; keine Autorisation
Akteure:	Beauftragter
Auslösendes Ereignis:	Fehlerhafter Eintrag

Beschreibung:	1 Authentifizierung 2 Suche des zu löschenden Eintrages 3 Eintrag ansehen 4 Löschen
Erweiterungen:	-
Alternativen:	1a bereits authentifiziert 2a Eintrag existiert nicht

Tabelle 07: Beschreibung des Geschäftsprozesses „Expeditionseintrag löschen“

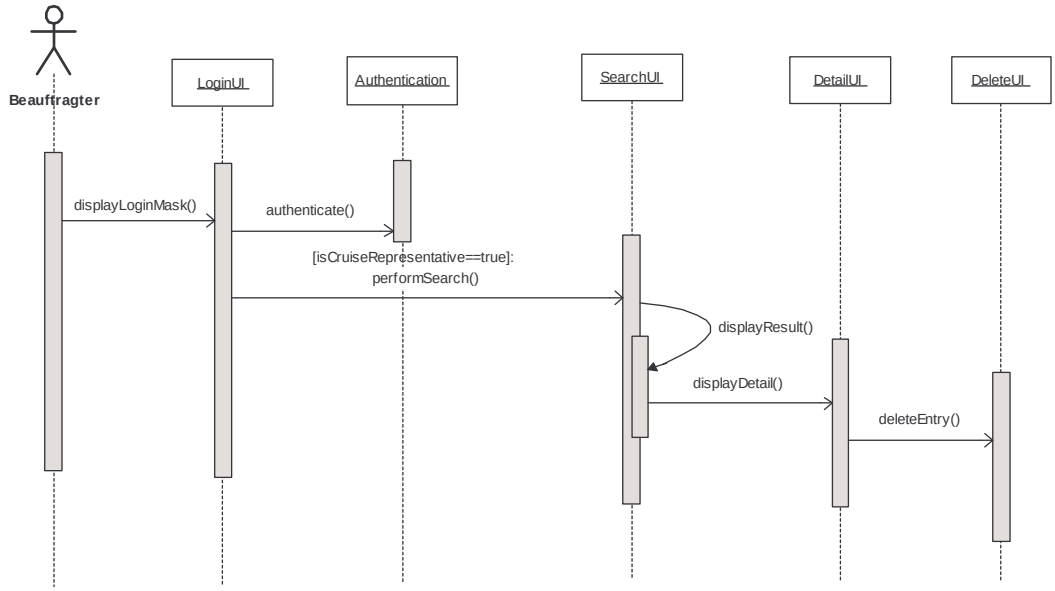

Abbildung 21: Sequenzdiagramm „Expeditionseintrag löschen“

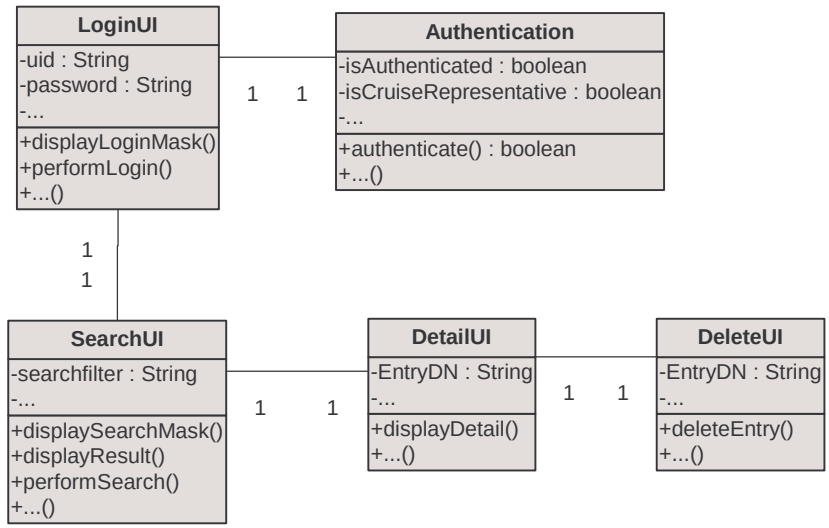

Abbildung 22: Klassendiagramm „Expeditionseintrag löschen“

7.2.4.2 Geschäftsprozess „Expeditionseintrag modifizieren“

Geschäftsprozess:	Expeditionseintrag modifizieren
Ziel:	Verwaltung von Expeditionsdaten, Expeditionseintrag modifizieren
Kategorie:	primär
Vorbedingung:	Eintrag existiert
Nachbedingung Erfolg:	Eintrag geändert
Nachbedingung Fehlschlag:	Eintrag nicht geändert; keine Autorisation
Akteure:	Beauftragter
Auslösendes Ereignis:	Eintrag ist nicht aktuell; fehlerhaft
Beschreibung:	1 Authentifizieren 2 Suchen des zu bearbeitenden Eintrages 3 Eintrag korrigieren
Erweiterungen:	-
Alternativen:	3a Eintrag löschen

Tabelle 08: Beschreibung des Geschäftsprozesses „Expeditionseintrag modifzieren“

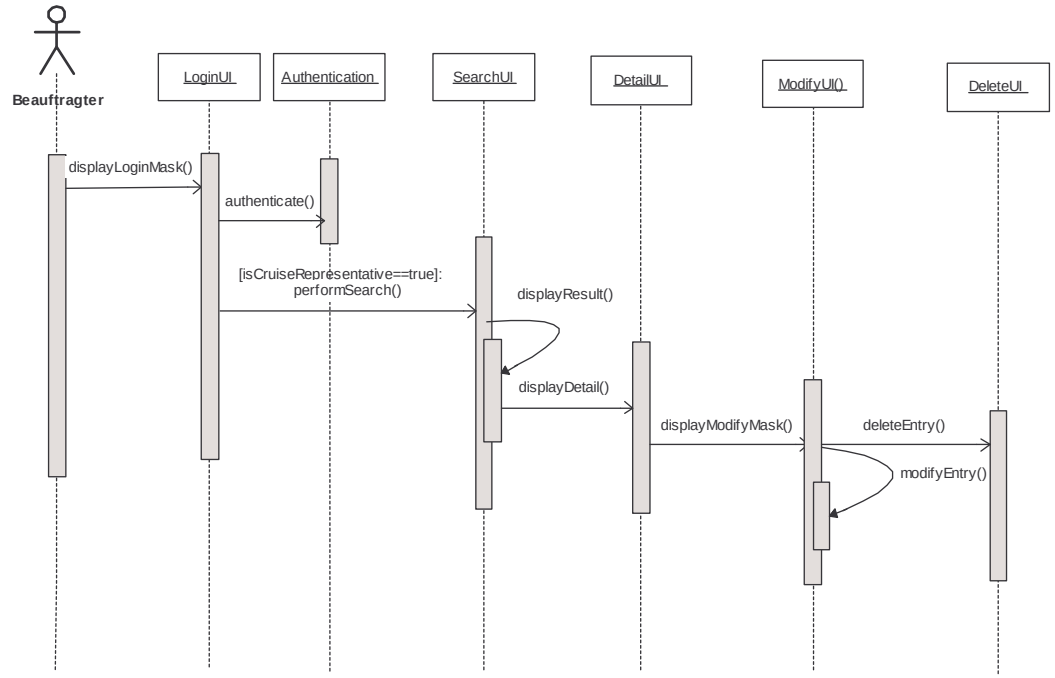

Abbildung 23: Sequenzdiagramm Expeditionseintrag „modifizieren“

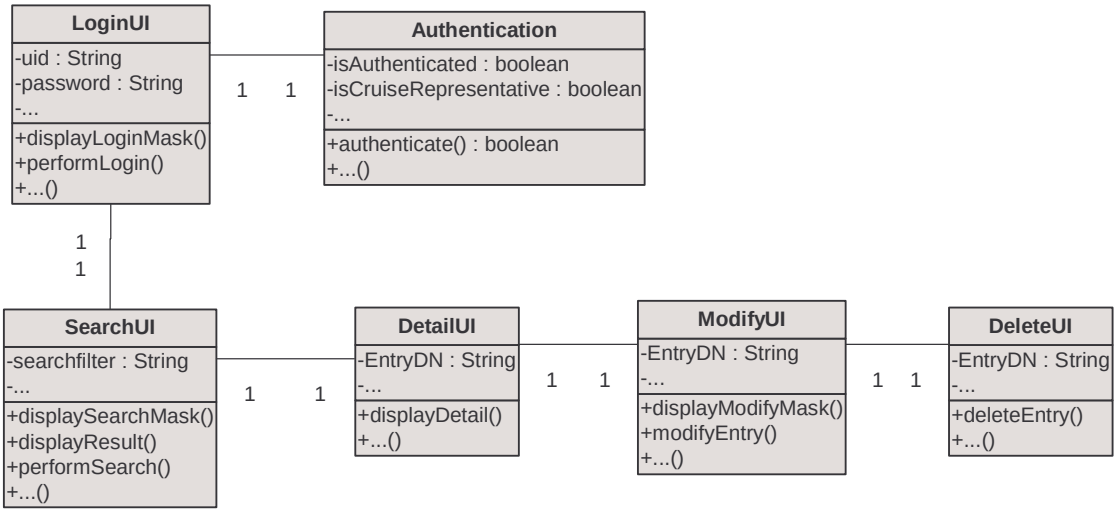

Abbildung 24: Klassendiagramm „Expeditionseintrag modifizieren“

7.2.4.3 Geschäftsprozess „Neuen Expeditionseintrag hinzufügen“

Geschäftsprozess:	Neuen Expeditionseintrag hinzufügen
Ziel:	Verwaltung von Expeditionsdaten, hinzufügen einer neuen Expedition
Kategorie:	primär
Vorbedingung:	Expedition ist noch nicht eingetragen
Nachbedingung Erfolg:	Eintrag erfolgreich hinzugefügt
Nachbedingung Fehlschlag:	Eintrag nicht hinzugefügt; keine Autorisation
Akteure:	Beauftragter
Auslösendes Ereignis:	Neue Expedition
Beschreibung:	1 Authentifizieren 2 Expeditionsdaten eingeben
Erweiterungen:	-
Alternativen:	-

Tabelle 09: Beschreibung des Geschäftsprozesses „Neuen Expeditionseintrag hinzufügen“

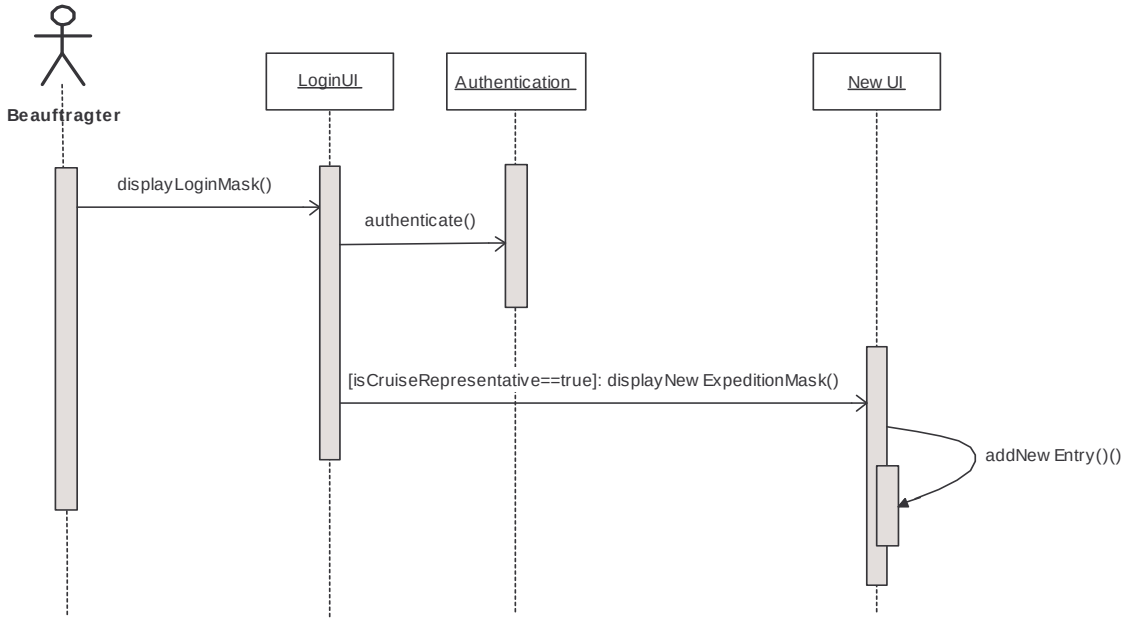

Abbildung 25: Sequenzdiagramm „Neuen Expeditionseintrag hinzufügen“

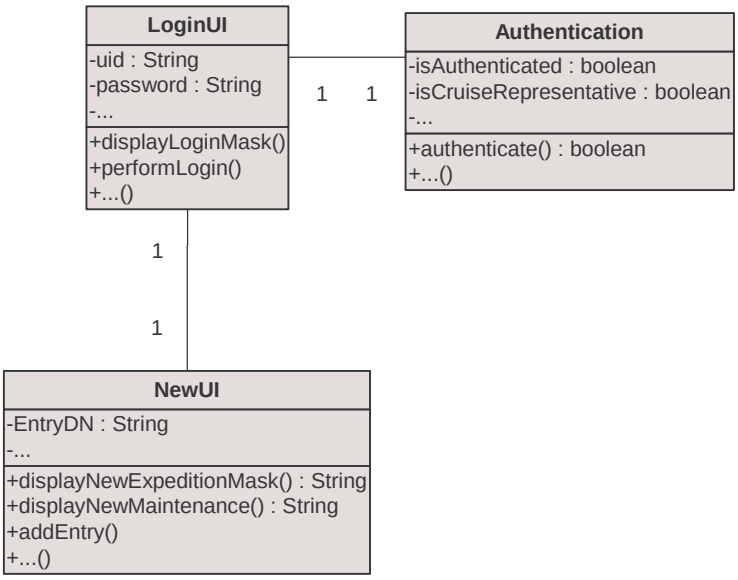

Abbildung 26: Klassendiagramm „Neuen Expeditionseintrag hinzufügen“

7.3 Entwurf des GUI-Systems

Aus dem OOA-Modell wird ein Prototyp der Benutzeroberfläche abgeleitet. Der Prototyp der Benutzeroberfläche zeigt die vollständige Oberfläche des zukünftigen Systems, ohne dass bereits Funktionalität realisiert ist.

Um den Prototyp der Benutzeroberfläche zu erstellen, muss zunächst ein Graphical User Interface (GUI) -System gewählt werden.

Ein (GUI) ist eine graphische Benutzeroberfläche. Sie besteht aus einer Dialogkomponente (Bedienungsabläufe) und einer Ein-/Ausgabe-Komponente (E/A-Komponente) (Gestaltung der Informationsdarstellung). Das GUI-System ist das Softwaresystem, das diese graphische Benutzeroberfläche verwaltet. Für die Erstellung des Prototyps wird idealerweise das gleiche GUI-System verwendet, das im Entwurf für die Realisierung der Benutzungsoberfläche benutzt wird.[9]

Da es sich bei der zu entwickelnden Anwendung um ein webbasiertes Expeditionsportal handelt, muss ein GUI-System gewählt werden, das sich zur Ausgabe und Darstellung in einem Webbrowser eignet. Die Wahl des GUI-Systems fällt deshalb auf die Hyper Text Markup Language (HTML).

HTML lässt sich auf jedem Browser darstellen. Es stellt Interaktions- und Dialogelemente zur Verfügung.

7.3.1 Navigation

Die Bedienung der Anwendung erfolgt über die HTML-typischen Dialog- und Interaktionselemente. Das sind im einzelnen Links, Absende-, Bestätigungs- und Abbruchbuttons.

Die Navigation zwischen den einzelnen Seiten der Anwendung, wird durch Links realisiert. Der Benutzer kann über das Anklicken der Links zu der gewünschten Funktionalität gelangen.

Als zentrale Navigationsstelle dient der Kopf der Anwendung, der Header. Im Kopf der Anwendung sind Links zu den zentralen Funktionen der Anwendung aufgeführt. So hat der Benutzer die Möglichkeit schnell und übersichtlich zwischen den einzelnen

[9] vgl. [Balzert, 1999] Seite 194 ff

Programmkomponenten zu wechseln.

Abbildung 27: Prototyp - Kopf der Anwendung – Navigationsleiste

Des weiteren erfolgt hier der Auswahl der Sprache, in der die Anwendung ausgegeben werden soll.

Die weitere Navigation erfolgt über HTML-Elemente wie Buttons, Formulare oder Pulldown-Menüs. Benutzereingaben werden durch die Betätigung von Buttons bestätigt oder abgebrochen. Meldungsfenster, die als Bestätigung von Eingaben oder als Fehlermeldungen ausgegeben werden, können durch Klicken von Bestätigungsbuttons verlassen werden.

7.3.2 Farben und Layout

Das Layout und die Farbgebung der Anwendung sind an die anderen AWI Webapplikationen angelehnt. Es existiert ein einheitliches Layout in Bezug auf die äußere Erscheinung der Anwendungen, um ein einheitliches Bild zu schaffen.

Das Layout und die Farbgebung sind in der gesamten Anwendung konsistent. Alle Seiten haben den gleichen Kopf. Die folgende Abbildung zeigt den Prototyp der Anwendung anhand der zukünftigen Startseite.

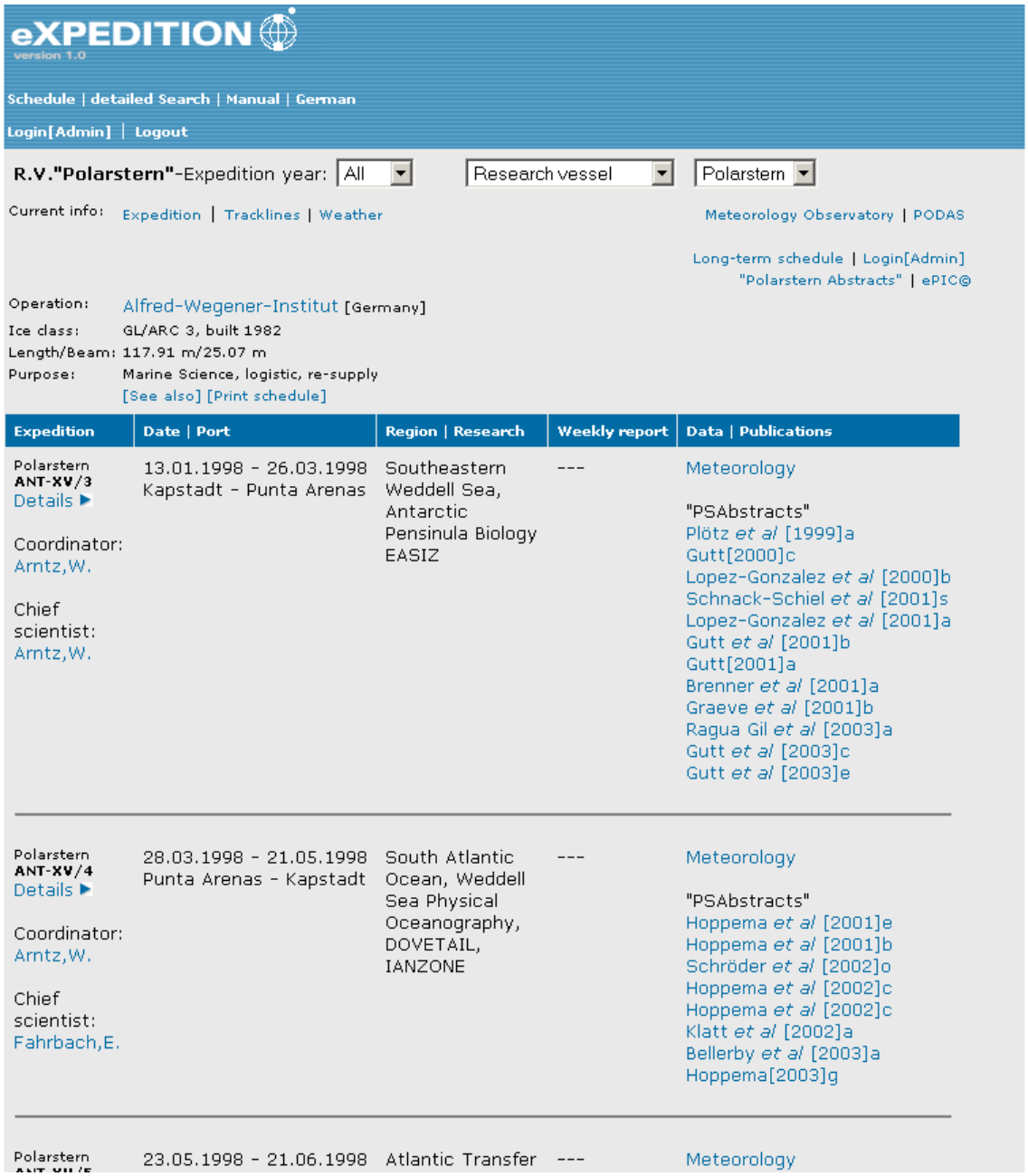

Abbildung 28: Prototyp – Startseite der Anwendung

7.3.3 Dialogstruktur der Anwendung

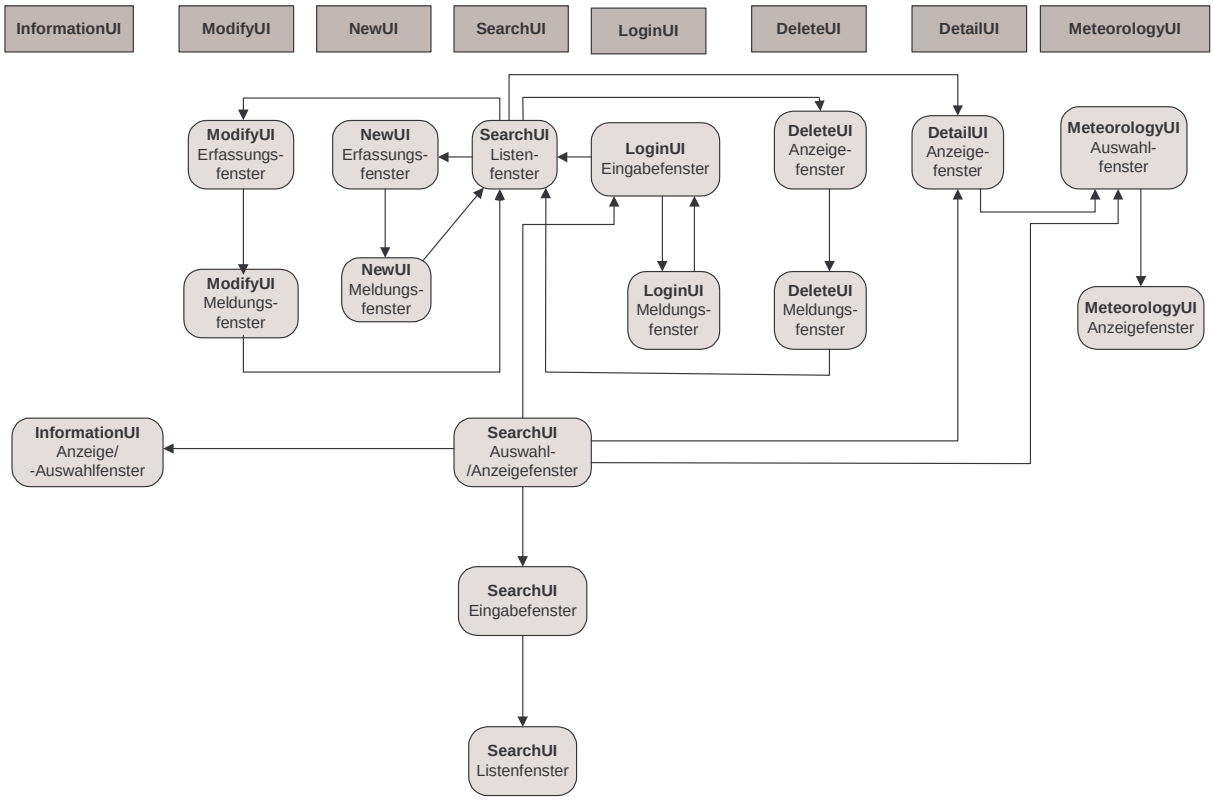

Abbildung 29: Zustandsdiagramm zur Modellierung der Dialogstruktur

Abbildung 29 zeigt die Dialogstruktur der Benutzeroberfläche der zu entwickelnden Anwendung. Dabei wird das Zusammenspiel der verschiedenen Ausgabefenster dargestellt. Die obere Zeile des Diagramms zeigt die für die jeweilige Ausgabe zuständige Klasse.

Von jeder Seite der Anwendung kann der Benutzer über die Navigationsleiste immer wieder zur Startseite zurück gelangen.

8 Objektorientierter Entwurf

In der Analyse wurde bei der Modellierung des Systems von einer idealen Umgebung ausgegangen. Aufgabe des Entwurfs ist es, die spezifizierte Anwendung auf einer Plattform unter den technischen Randbedingungen zu realisieren.

8.1 OOD-Modell

Das Ergebnis des Entwurfs ist das OOD-Modell. Das OOD-Modell ist die technische Lösung des zu realisierenden Systems, die in einer objektorientierten Notation modelliert wird. Das OOD-Modell ist ein Abbild des späteren objektorientierten Programms.

8.2 Architekturentwurf

Während des Architekturentwurfs wird die Systemarchitektur erstellt, d.h. es werden die grundlegenden Komponenten des Softwaresystems und die Abhängigkeiten zwischen ihnen dargestellt. Durch die Architektur wird die Anwendung in Schichten (layers, tiers) unterteilt.[10]

Zwischen den Schichten gelten strenge Zugriffsregeln.

8.2.1 Zwei-Schichten-Architektur

Die Zwei-Schichten-Architektur (two-tier architecture) besteht aus einer Anwendungsschicht, die die Benutzeroberfläche und das Fachkonzept enthält, und einer Datenhaltungsschicht.

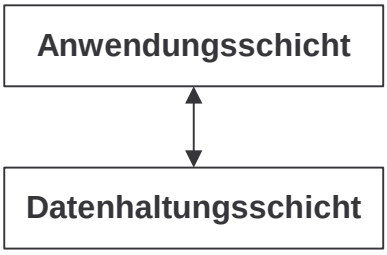

Abbildung 30: Zwei-Schichten-Architektur

8.2.2 Drei-Schichten-Architektur

Die Drei-Schichten-Architektur (three-tier architecture) besteht aus einer GUI - Schicht, der Fachkonzeptschicht und der Datenhaltungsschicht. Die GUI-Schicht realisiert die Benutzeroberfläche einer Anwendung. Dazu gehören die Dialogführung und die Präsentation aller Daten in Fenstern, Berichten, etc.

[10] vgl. [Balzert, 1999]

Die Fachkonzeptschicht modelliert den funktionalen Kern der Anwendung. Des weiteren enthält sie die Zugriffe auf die Datenhaltungsschicht.
In der Datenhaltungsschicht wird die jeweilige Form der Datenspeicherung realisiert, z.B. mit einem Datenbanksystem.

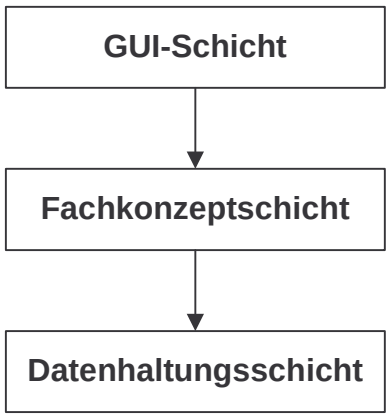

Abbildung 31: Drei-Schichten-Architektur

Bei der Drei-Schichten-Architektur sind zwei Ausprägungen möglich. Bei einer strengen Drei-Schichten-Architektur kann die GUI-Schicht nur auf die Fachkonzeptschicht und letztere nur auf die Datenhaltungsschicht zugreifen.
Eine flexible Drei-Schichten-Architektur ergibt sich, wenn die GUI-Schicht nicht nur auf die Fachkonzeptschicht, sondern zusätzlich auf die Datenhaltungsschicht zugreifen kann. [11]

8.2.3 Schichtenarchitektur der Anwendung

Das zu entwickelnde System weist eine Drei-Schichten-Architektur mit strenger Ordnung auf. Die einzelnen Programmkomponenten lassen sich eindeutig den verschiedenen Schichten zuordnen.
Auf der GUI-Schicht befinden sich die Programmkomponenten, die für die Ausgabe der Inhalte verantwortlich sind. Sie nehmen Benutzereingaben auf und stellen die Benutzeroberfläche dar. Die Programmkomponenten auf der Fachkonzeptschicht sind zuständig für die Anwendungslogik. Sie verarbeiten die Benutzereingaben, bereiten die Daten für die weitere Verarbeitung, z.B. das Eintragen in den Directory Server, vor und bereiten die Daten für die Ausgabe auf. Des weiteren stellen sie die Methoden bereit,

[11] vgl. [Balzert, 1999]

die Operationen für den Zugriff auf die Daten im Directory Server, Datenbank und den Dateiverzeichnisse ermöglichen.

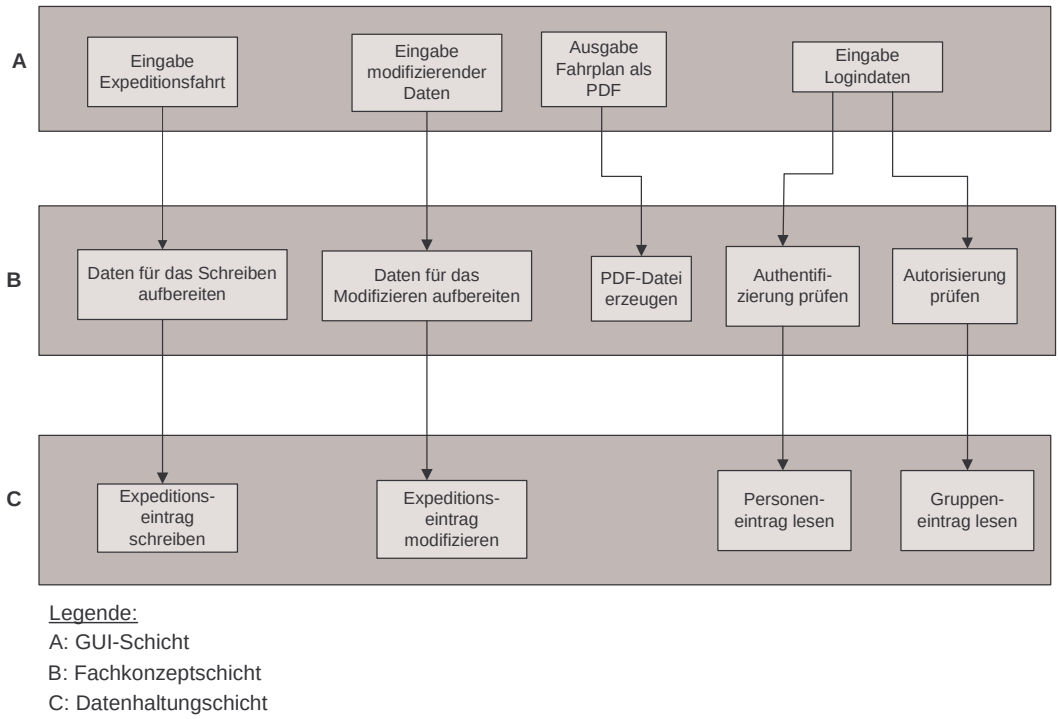

Abbildung 32: Verteilung und Zusammenspiel der Programmkomponenten auf den einzelnen Schichten

Anhand der Abbildung 32 wird beispielhaft an einigen typischen Anwendungsfällen die Verteilung und das Zusammenspiel der einzelnen Programmkomponenten auf den einzelnen Schichten dargestellt.

Daraus wird deutlich, dass es sich um eine Drei-Schichten-Architektur mit strenger Ordnung handelt. Es gibt keine Zugriffe über mehr als eine Schicht. Jede Schicht kann nur auf die jeweils darunter liegende Schicht zugreifen. So hat z.B. die GUI-Schicht nur Zugriff auf die darunter liegende Fachkonzeptschicht und nicht direkt auf die Datenhaltungsschicht.

8.3 Programmstruktur

Im folgenden wird die Struktur der zukünftigen Anwendung und der im OOD entworfenen Klassen beschrieben.

8.4 Paketstruktur der entworfenen Klassen

Die entworfenen Klassen werden in Paketen organisiert.

Die Struktur der gebildeten Pakete orientiert sich an der Paketstruktur der bereits vorhandenen Java-Anwendungen ePIC und ePersonal. Der Entwurf der Anwendung umfasst die folgenden Pakete

- awi.expedition, beinhaltet die Klassen, die für die Anwendungslogik zuständig sind.
- awi.expedition.ui, umfasst die UI-Klassen, die für die Ausgabe zuständig sind.
- awi.util, enthält Klassen, die nützliche Methoden, z.B. Operationen mit Strings, bereitstellen.

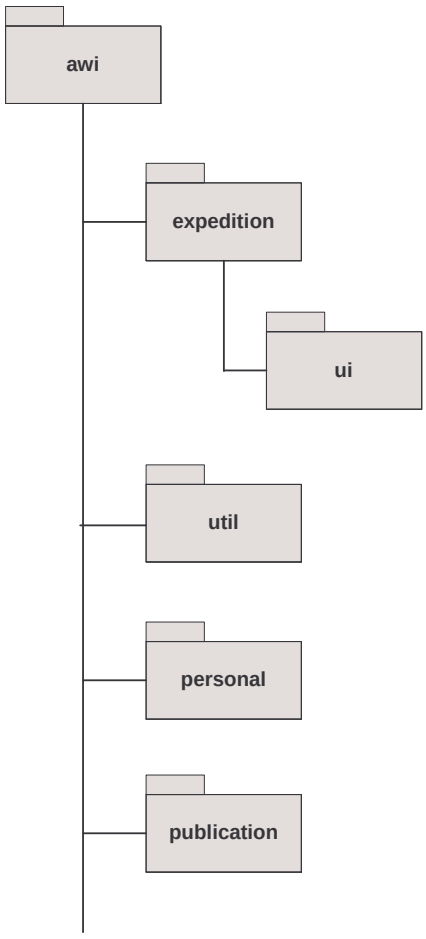

Abbildung 33: Paketstruktur

8.5 Verwendete Klassen der bestehenden AWI Java-Anwendungen

Es werden Klassen der bereits bestehenden AWI Java-Anwendungen ePIC und ePersonal benutzt. Diese Anwendungen stellen schon ein Rahmenwerk, um auf Publikationseinträge und Personeneinträge zuzugreifen. Nachfolgend werden die benutzten Klassen dieser Anwendungen und die für diese Anwendung relevanten Methoden beschrieben

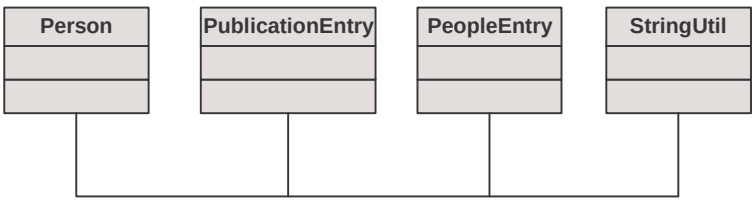

Abbildung 34: Benutzte Klassen der bestehenden AWI-Anwendungen

8.5.1 ePersonal

Diese Anwendung stellt Möglichkeiten bereit um komfortabel auf Personeneinträge und Personendaten der im Directory Server abgelegten AWI Mitarbeiter zuzugreifen. Für die zu entwickelnde Anwendung sind hierbei u.a. Daten über die uid, den Vornamen und den Nachnamen einer Person wichtig.

8.5.1.1 Beschreibung der Klasse PeopleEntry

Die Klasse PeopleEntry spiegelt einen Personeneintrag im Directory Server wieder. Man erhält Zugriff auf alle Attribute eines Personeneintrages. Die Klasse enthält Methoden, mit denen auf die einzelnen Attribute zugegriffen werden kann. Die Werte können ausgelesen und geändert werden. Für die Expeditionsanwendung sind nur die Methoden relevant mit denen auf die einzelnen Attribute zugegriffen werden kann. Diese Methoden haben den Namen *getAttributname()*. Die verwendeten Methoden werden in der folgenden Tabelle aufgeführt.

Klasse	**PeopleEntry**
Attribut	private String firstname;
	private String lastname;

	private String name;
	private String mobile;
	private String phone;
	private String fax;
	…
Methode	PeopleEntry()
	PeopleEntry(PeopleEntry People)
	public String getFirstname ()
	public String getLastname ()
	public String getPassword ()
	…

Tabelle 10: Klasse PeopleEntry

8.5.1.2 Beschreibung der Klasse Person

Die Klasse Person repräsentiert eine Person am AWI. Dies beinhaltet alle Attribute die über die Person in ihrem Personeneintrag im Directory Server gespeichert sind, sowie alle weiteren Informationen, die über die Person im Directory Server und anderen Datenquellen vorhanden sind. Dazu gehören die Zugehörigkeit der Person zu den verschiedenen Fachbereichen und Gruppen sowie die Publikationen, die mit der Person in Bezug stehen. Für die Expeditionsanwendung wird diese Klasse benutzt, um einfach auf die Personendaten zuzugreifen. In der Session wird ein Person-Objekt abgelegt, anhand dessen schnell wieder auf die benötigten Personendaten wie das Passwort oder die uid zugriffen werden kann, wenn sie benötigt werden, z.B. zum vorherigen Authentifizieren bevor ein neuer Expeditionseintrag geschrieben wird.

Klasse	**Person**
Attribut	...
Methode	Person ()
	Person(LDAPConnection ld, String dn)
	…

Tabelle 11: Klasse Person

8.5.2 Epic

8.5.2.1 Beschreibung der Klasse PublicationEntry

Die Klasse PublicationEntry bildet einen Eintrag eines Publikationseintrages im Directory Server ab. Man erhält Zugriff auf die einzelnen Attribute eines Publikationseintrages. Über die Methoden *getAttribut()* erhält man die Werte der einzelnen Attribute. Die für die Anwendung relevanten Methoden werden in der folgenden Tabelle aufgeführt.

Klasse	**PublicationEntry**
Attribut	private Vector uid;
	private Vector publicationauthor;
	private Vector publication_email
	private Vector publication_title
	private Vector publication_citation
	private Vector publication_year
	…
Methode	PublicationEntry()
	PublicationEntry(String entryDN)
	public Vector get_publication_authors()
	public Vector get_uid()
	public Vector get_publication_year()
	...

Tabelle 12: Klasse PublicationEntry

8.5.2.2 Beschreibung der Klasse StringUtil

Die Klasse StringUtil stellt Methoden bereit um nützliche Operationen mit Strings durchzuführen. Dazu gehören u.a. einen Teilstring in einem String durch einen anderen String zu ersetzen.

Klasse	**StringUtil**
Attribut	...
Methode	StringUtil()

	public String kill_linebreaks(String original_String)
	public String string_replace(String targetString, String replaceThis, String byThis)
	public String[] string_split(String targetString, String separator)

Klasse StringUtil

8.6 Klassendiagramm und –struktur

Nachfolgend werden die für die Anwendung entworfenen Klassen und ihre Struktur beschrieben.

Als Einstieg für die Anwendung dient ein einziges zentrales Servlet. Dieses Servlet, steuert über Parameter, welche andere Klasse benötigt wird und lädt diese. Neben dem Servlet existiert eine Anzahl von entworfenen Klassen, die für die Anwendungslogik und die Benutzerinteraktion zuständig sind.

Die Klassen, die für die Ausgabe der Benutzeroberfläche und für die Benutzerinteraktion zuständig sind, werden in dem Paket awi.expedition.ui zusammengefasst. Diese Klassen haben den Suffix „UI“ im Namen. UI ist dabei die Abkürzung für User Interface.

Das nachfolgende Diagramm zeigt die Klassenstruktur der entworfenen Klassen auf. Dabei wurde aus Gründen der besseren Lesbarkeit auf die Aufführung von Attributen und Methoden der einzelnen Klassen verzichtet. Bei der Klasse display.java handelt es sich um das Servlet, das die Anwendung steuert. Alle UI-Klassen erben von der Klasse GeneralUI, die für alle UI-Klassen wiederkehrende Methoden implementiert und generelle Designvorgaben beinhaltet.

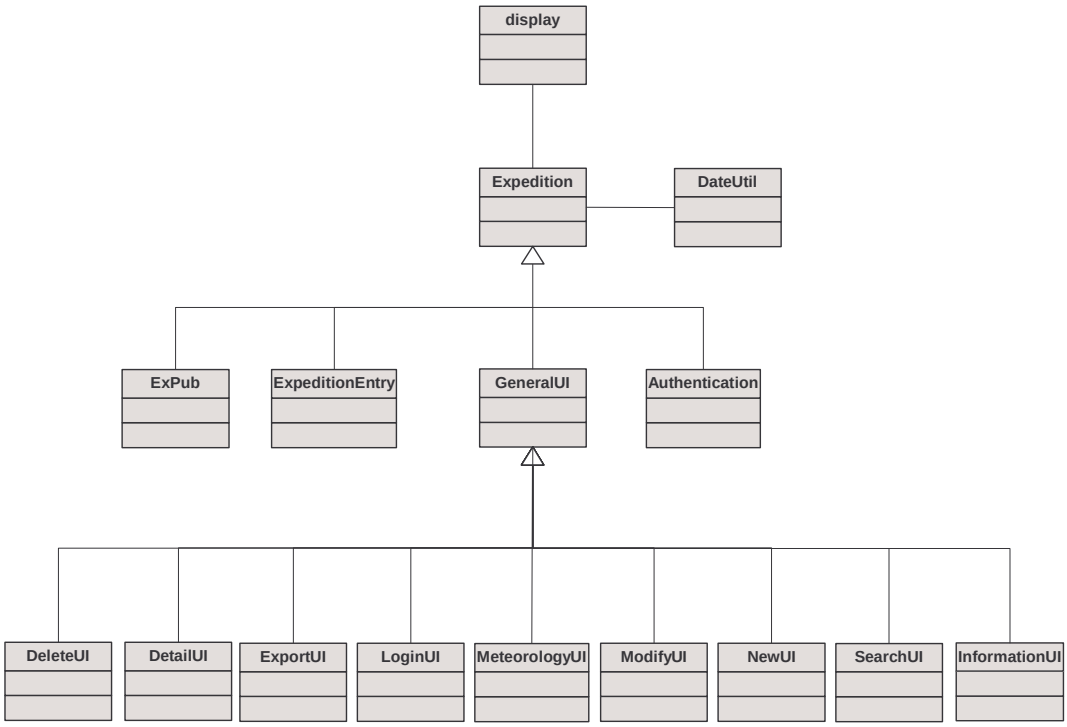

Abbildung 35: Klassendiagramm der entworfenen Klassen

8.7 Beschreibungen der einzelnen Klassen

Nachfolgend werden die im OOD entworfenen Klassen beschrieben. Die entworfenen Klassen lassen sich aufteilen in Klassen, die für die Ausgabe zuständig sind und in Klassen, die für die Programmlogik zuständig sind.

8.7.1 Beschreibung des Servlets display

Die Anwendung wird vom dem Servlet display.java aufgerufen und gesteuert. In ihm werden die Request-Anfragen vom Client und die Response-Antworten des Servers über die Methoden *doGet()* und *doPost()* bearbeitet. Das Servlet setzt den ContentType des generierten Inhaltes. Der ContentType gibt an, um welche Art von Dokument es sich handelt. Der ContentType ist für die Ausgabe der dynamisch generierten Inhalte „text/html“. Über die Abfrage von Parametern wird entschieden, welche Klasse und welche Methoden geladen werden sollen. Das Servlet implementiert Standardmethoden, die den Lebenszyklus des Servlets steuern.

Klasse	**destroy**

Attribut	...
Methode	public display()
	public void destroy()
	public void doGet(HtttpRequest request, HttpResponse response)
	public void doPost(HtttpRequest request, HttpResponse response)
	public String getServletInfo()
	public void init(ServletConfig sc)

Tabelle 13: Klasse destroy

8.7.2 Beschreibung der Klasse Expedition

In der Klasse Expedition werden einige statische Variablen definiert. Diese Variablen beinhalten z.B. Pfadangaben, Host- oder Port-Angaben. Die Variablen erhalten ihre Werte aus der Properties-Datei „expedition.properties." Die so zugewiesenen Werte sind Konstanten, sind statisch, d.h. sie können während des Programmablaufs nicht geändert werden. Alle anderen Klassen erben von dieser Klasse. Die Klassen haben so Zugriff auf die hier definierten statischen Variablen.

Klasse	**Expedition**
Attribut	final protected static String HOST;
	final protected static int PORT;
	final protected static String ROOT_DN;
	final protected static String PEOPLE_BRANCH;
	final protected static String GROUPS_BRANCH;
	final protected static String CRUISE_BRANCH;
	final protected static String PUBLICATION_BRANCH;
	...
Methode	public Expedition()

Tabelle 14: Klasse Expedition

8.7.3 Beschreibung der Klasse Authentication

In der Klasse Authentication werden Methoden bereitgestellt, die dazu dienen eine Person zu authentifizieren und Autorisierungen zu überprüfen.

Die Methode *authenticate(String uid, String password)* authentifiziert die übergebene UID mit dem übergebenen Passwort gegenüber dem Directory Server. Diese Methode liefert als Rückgabewert einen String mit der eventuell auftretenden Fehlermeldung. Die Methode *checkCruiseRepresentative(String uid)* prüft, ob die übergebene UID, der Benutzer, die Autorisation als Representative, d.h. als Administrator für die Anwendung hat.

Klasse	**Authentication**
Attribut	...
Methode	public Authentication()
	public String authenticate(String uid, String password)
	public boolean checkCruiseRepresentative(String uid)
	public String return_authMessage()
	public boolean return_CruiseRepresentative()

Tabelle 15: Klasse Authentication

8.7.4 Beschreibung der Klasse ExpeditionEntry

Die Klasse ExpeditionEntry spiegelt einen Expeditionseintrag im Directory Server wieder. Sie implementiert Methoden, die Zugriffe und Operationen auf den Directory Server realisieren. Des weiteren werden Methoden bereitgestellt, die Dateien aus Verzeichnissen auf dem Server auslesen. Die Klasse ExpeditionEntry erbt von der Klasse Expedition und implementiert die Klasse FilenameFilter.

Die Methode *createNewExpeditionEntry(HttpServletRequest request, HttpServletResponse response)* definiert ein LDAPAttributeSet mit den Attributen und Werten, die der zukünftige Eintrag haben soll und schreibt diesen in den Directory Server.

Die Methode *modifyEntry(HttpServletRequest request, HttpServletResponse response, String CruiseID)* ändert den Eintrag der übergebenen CruiseID im Directory Server. Dabei wird ein LDAPModificationSet definiert, in dem die Attribute mit den neuen

Werten hinzugefügt werden. Anhand dieses LDAPModificationSet wird der entsprechende Eintrag im Directory Server geändert.

Die Methode *deleteEntry(HttpServletRequest request, HttpServletResponse response, String CruiseID)* löscht den Eintrag der übergebenen CruiseID aus dem Directory Server.

Die Methoden *getCruiseSummary(...), getMap(...) und getNewsletter(...)* durchsuchen in den angegebenen Verzeichnissen auf dem Server, ob für die übergebene Expedition Fahrtzusammenfassungen, Karten oder Newsletter vorhanden sind.

Die Methode *getEntry(LDAPConnection ld, String dn)* holt für den Eintrag des übergebenen DN die Attribute und Werte aus dem Directory Server. Die erhaltenen Werte werden über die Methode *setVectorValues (LDAPAttribute attribute)* in Vektoren gespeichert und über Methoden mit den Namen *return_Attributname* zurückgegeben.

Klasse	**ExpeditionEntry**
Attribut	private Vector cruiseID;
	private Vector cruisebeginn;
	private Vector cruiseend;
	private Vector cruiseportbeginn;
	private Vector cruiseportend;
	private Vector cruisecaptain;
	…
Methode	public ExpeditionEntry()
	public boolean accept(File directory, String name)
	public void createNewExpeditionEntry(HttpServletRequest request, HttpServletResponse response)
	public void createNewMaintenanceEntry(HttpServletRequest request, HttpServletResponse response)
	public void deleteEntry(HttpServletRequest request, HttpServletResponse response, String CruiseID)
	public void FillModificationSet_Delete(LDAPModificationSet modSet, String attribute_name, String attributevalue)

	public void FillModificationSet_Replace(LDAPModificationSet modSet, String attribute_name, String attributevalue)
	public void getCruiseSummary(String CruiseID)
	public void getEntry(LDAPConnection ld, String dn)
	public void getMap(String CruiseID)
	public void getNewsletter(String CruiseID)
	public void getUid (LDAPConnection ld, String uid)
	public void modifyEntry(HttpServletRequest request, HttpServletResponse response, String CruiseID)
	public void setVectorValues (LDAPAttribute attribute)
	public Vector return_cruiseID()
	public Vector return_cruiseregion()
	public Vector return_cruiseresearch()
	…

Tabelle 16: Klasse ExpeditionEntry

8.7.5 Beschreibung der Klasse DateUtil

In der Klasse DateUtil werden Methoden bereitgestellt, die Operationen mit einem Datum durchführen. Diese Methoden sind speziell für die zu entwickelnde Anwendung und werden in der java.util.Date nicht bereitgestellt.

Dazu gehören die Konvertierung, Sortierung und das Vergleichen von Daten.

Die Methode *formatStringtoDate(String d)* beispielsweise konvertiert ein Datum, das in einem String vorliegt, in ein Datum mit dem Format Date.

Die Methode *sortResultsByDate(Vector DNs, LDAPConnection ld)* sortiert die bei einer Suche erhaltenen Suchergebnisse nach dem Anfangsdatum.

Klasse	**DateUtil**
Attribut	...
Methode	public DateUtil()
	public boolean compareDateMeteorology(String beginDate, Date today)

	public Boolean compareDate(String beginnDate, String endDate, Date today)
	public Date formatStringtoDate(String d)
	public String getCurrentYear_long()
	public String getCurrentYear_short()
	public Vector sortResultsByDate(Vector DNs, LDAPConnection ld)
	private Comparator getComparator(final int index, final boolean reverse)

Tabelle 17: Klasse DateUtil

8.7.6 UI-Klassen

Im folgenden werden die entworfenen Klassen, die zur Ausgabe und Darstellung des dynamisch genierten Inhaltes und der HTML-Elemente dienen, beschrieben.

8.7.7 Beschreibung der Klasse GeneralUI

Die Klasse GeneralUI implementiert einige Methoden, die bei der Ausgabe häufig benutzt werden, z.B. die Ausgabe von Fehlermeldungen. Des weiteren werden in ihr generelle Designvorgaben getroffen. Alle UI-Klassen erben von der GeneralUI. GeneralUI ist die Oberklasse für alle UI-Klassen, die somit Zugriff auf die dort implementierten Methoden haben.

Die Methode *displayExpeditionHeader(HttpServletRequest request, HttpServletResponse response)* gibt den Header der Anwendung aus. Die Methode *displayErrorMessage(HttpServletRequest request, HttpServletResponse response, String message)* ist für die Ausgabe von Fehlermeldungen zuständig.

Klasse	**GeneralUI**
Attribut	...
Methode	public GeneralUI()
	public void diplayButtonBack(HttpServletRequest request, HttpServletResponse response)
	public void displayErrorMessage(HttpServletRequest request, HttpServletResponse response, String message)

	public void displayExpeditionHeader(HttpServletRequest request, HttpServletResponse response)
	public void displayFooter(HttpServletRequest resquest, HttpServletResponse response)
	public void displayLinkBacktoOverview(HttpServletRequest request, HttpServletResponse response)

Tabelle 18: Klasse GeneralUI

8.7.8 Beschreibung der Klasse DeleteUI

Die Klasse DeleteUI ist eine Unterklasse der Klasse GeneralUI.

Sie zeigt den zu löschenden Eintrag an und stößt das Löschen nach Absenden des Formulars an.

Klasse	**DeleteUI**
Attribut	...
Methode	public DeleteUI()
	public void displayDeleteMask(HttpServletRequest request, HttpServletResponse response)

Tabelle 19: Klasse DeleteUI

8.7.9 Beschreibung der Klasse DetailUI

Die Klasse DetailUI generiert dynamisch die Detailansicht für die einzelnen Expeditionsfahrten und Wartungseinträge.

Die Methode *displayDetail()* gibt die Details für die übergebene CruiseID aus.

Klasse	**DetailUI**
Attribut	...
Methode	public DetailUI()
	public void displayDetail(HttpServletRequest request, HttpServletResponse response, String CID)
	public void displayMaintenanceDetail(HttpServletRequest request, HttpServletResponse response, String CID)
	public String extractTitleFromHTMLFile(String path)

Tabelle 20: Klasse DetailUI

8.7.10 Beschreibung der Klasse ExportUI

In dieser Klasse wird dynamisch eine PDF-Datei für das übergebene Forschungsschiff und das Expeditionsjahr generiert.

Klasse	**ExportUI**
Attribut	...
Methode	public ExportUI()
	public void generatePDF(HttpServletRequest request, HttpServletResponse response, String ShipName, String year)

Tabelle 21: Klasse ExportUI

8.7.11 Beschreibung der Klasse ExPub

In dieser Klasse werden die zu der jeweiligen Expeditionsfahrt gehörenden AWI-Publikationen, CruiseReports und PSAbstracts ermittelt.

Klasse	**ExPub**
Attribut	...
Methode	public ExPub()
	public boolean accept(File directory, String name)
	public String charakter(String Charakter, String aString)
	public Vector getAbstractsFieldReport(HttpServletRequest request, HttpServletResponse response, Vector PubDn)
	public Vector getAuthorsPSAbstracts(HttpServletRequest request, HttpServletResponse response, Vector PubDn)
	public Vector getPSAbstracts(String Cruiseid)

Tabelle 22: Klasse ExPub

8.7.12 Beschreibung der Klasse InformationUI

Die Klasse InformationUI gibt die speziellen Informationen zu den einzelnen Forschungseinrichtungen aus. Das können Informationen zu allen in der Anwendung eingebundenen Forschungseinrichtungen sein.

Klasse	**InformationUI**

Attribut	...
Methode	public InformationUI()
	public void displayInformationAircraft(HttpServletRequest request, HttpServletResponse response, String name)
	public void displayInformationLandStation(HttpServletRequest request, HttpServletResponse response, String name)
	public void displayInformationOceanStation(HttpServletRequest request, HttpServletResponse response, String name)
	public void displayInformationShip(HttpServletRequest request, HttpServletResponse response, String name)

Tabelle 23: Klasse InformationUI

8.7.13 Beschreibung der Klasse LoginUI

Die Klasse LoginUI gibt die Login-Maske aus. Die darin eingegebenen Daten werden für die weitere Verarbeitung zur Authentifizierung und Autorisierung vorbereitet. Die Methode *displayLoginMask(HttpServletRequest request, HttpServletResponse response)* gibt die Loginmaske aus.

Klasse	**LoginUI**
Attribut	...
Methode	public LoginUI()
	public void displayLoginMask(HttpServletRequest request, HttpServletResponse response)
	public void performLogin(HttpServletRequest request, HttpServletResponse response)

Tabelle 24: Klasse LoginUI

8.7.14 Beschreibung der Klasse MeteorologyUI

Die Klasse MeteorologyUI gibt eine Abfragemaske mit verschiedenen Kriterien, die aus einer Datenbank mit meteorologischen Daten abgefragt werden können, aus.

Die Methode getMeteorologyData(HttpServletRequest request, HttpServletResponse response) holt die jeweiligen meteorologischen Daten aus der Datenbank.

Klasse	**MeteorologyUI**
Attribut	...
Methode	public MeteorologyUI()
	public void displayMeteorologyMask(HttpServletRequest request, HttpServletResponse response, String CruiseID)
	public void getMeteorologyData(HttpServletRequest request, HttpServletResponse response)

Tabelle 25: Klasse MeteorologyUI

8.7.15 Beschreibung der Klasse ModifyUI

Die Klasse ModifyUI gibt die Maske zur Änderung der Daten eines Eintrages aus. Dabei werden die Felder der Maske mit den alten Werten des zu verändernden Eintrages vorbelegt. Die zu ändernden Daten werden für die weitere Verarbeitung vorbereitet.

Klasse	**ModifyUI**
Attribut	...
Methode	public ModifyUI()
	public void displayModifyExpeditionMask(HttpServletRequest request, HttpServletResponse response, String CID)
	public void displayModifyMaintenanceMask(HttpServletRequest request, HttpServletResponse response, String CID)
	public void perform(HttpServletRequest request, HttpServletResponse response)
	public void prepareValues (HttpServletRequest request, HttpServletResponse response)

Tabelle 26: Klasse ModifyUI

8.7.16 Beschreibung der Klasse NewUI

Die Klasse NewUI stellt die Masken zur Eintragung neuer Expeditionseinträge und Wartungseinträge dar. Außerdem werden die eingegebenen Daten für den neuen Eintrag für die weitere Verarbeitung aufbereitet.

Klasse	**NewUI**
Attribut	...
Methode	public NewUI()
	public void displayNewExpeditionMask(HttpServletRequest request, HttpServletResponse response)
	public void displayNewMaintenanceMask(HttpServletRequest request, HttpServletResponse response)
	public void perform(HttpServletRequest request, HttpServletResponse response)
	public void prepareMaintenanceValuesToAdd(HttpServletRequest request, HttpServletResponse response)
	public void prepareValuesToAdd(HttpServletRequest request, HttpServletResponse response)

Tabelle 27: Klasse NewUI

8.7.17 Beschreibung der Klasse SearchUI

Die Klasse SearchUI führt die benötigten Suchen auf dem Directory Server durch. Die Suchergebnisse werden verarbeitet, aufbereitet und entsprechend ausgegeben. Des weiteren werden hier die Ausgaben für die Suchmaske, den Zeitplan und die Suchergebnisse erzeugt.

Klasse	**SearchUI**
Attribut	...
Methode	public SearchUI()
	public void displayOverview(HttpServletRequest request, HttpServletResponse response)
	public void displayResult(HttpServletRequest request, HttpServletResponse response)
	public void displaySchedule(HttpServletRequest request, HttpServletResponse response)
	public void displaySearchMask(HttpServletRequest request, HttpServletResponse response)

	public void displaySelectionMenuAircraft(HttpServletRequest request, HttpServletResponse response)
	public void displaySelectionMenuLandStation(HttpServletRequest request, HttpServletResponse response)
	public void displaySelectionMenuOceanStation(HttpServletRequest request, HttpServletResponse response)
	public void displaySelectionMenuShip(HttpServletRequest request, HttpServletResponse response)
	public void partialOverview(int Seite, HttpServletRequest request, HttpServletResponse response)

Tabelle 28: Klasse SearchUI

9 Implementierung

Die Implementierung beschreibt die Umsetzung der im OOD entworfenen Klassen mit der gewünschten Programmiersprache. In diesem Fall wurde die Programmiersprache Java gewählt. Da es sich um eine webbasierte Anwendung handelt wird eine API benötigt, mit der serverseitige Programme erstellt und dynamische Webseiten generiert werden können. Dazu wird die Servlet API eingesetzt.

9.1 Verwendete Bibliotheken

Im folgenden werden die zusätzlichen Java Bibliotheken beschrieben, die zur Realisierung des entworfenen Programms benutzt wurden und nicht zum Standardlieferumfang des JDKs gehören.

9.1.1 Java Servlet API

Die Java Servlet API wird benutzt um serverseitige Javaprogramme zu schreiben mit denen dynamische Webseiten erstellt werden können. Die Servlet API beinhaltet die Pakete javax.servlet und javax.servlet.http.

9.1.2 PDFLib

PDFLib ist eine Bibliothek, mit der Dateien im Portable Document Format (PDF) erstellt werden können. PDFLib wird hauptsächlich dazu verwendet, um aus eigener Software heraus dynamisch PDF zu generieren. PDFLib kann unmittelbar in die Anwendung, die für die Datengenerierung zuständig ist, integriert werden. Bei der Programmierung der Anwendung wird die PDFLib dazu benutzt, die dynamisch generierten Zeitpläne der Forschungsschiffe als PDF-Datei zu exportieren.

9.1.3 LDAP-SDK

Das Netscape LDAP-SDK bietet die Möglichkeit mit Java auf Netscape Directory Server zuzugreifen. Es stellt Methoden zur Verfügung mit denen es möglich ist, auf LDAP Directory Server zuzugreifen. Die bereitgestellten Methoden bieten die Möglichkeit das Verzeichnis zu durchsuchen, neue Einträge hinzuzufügen, Einträge zu modifizieren und Einträge zu löschen. Eine alternative Zugriffsmöglichkeit auf den Directory Server wäre Java Naming and Directory Interface (JNDI). Allerdings ist das LDAP SDK auf den LDAP-Zugiff spezialisiert und bietet mehr Funktionen an als JNDI mit dem entsprechenden LDAP-Zugriff.

9.1.3.1 LDAP Connection

Um auf den Directory Server zuzugreifen, muss zunächst eine Verbindung mit ihm hergestellt werden.

Über den Befehl

```
LDAPConnection ld = new LDAPConnection();
```

wird ein neues Objekt einer LDAP-Verbindung erzeugt.

Durch Aufruf der Methode

```
ld.connect(String HOST, int PORT);
```

wird eine Verbindung zu einem LDAP Server unter Angabe des Host-Namens und des Ports hergestellt. Die Methode

```
ld.disconnect();
```

beendet die Verbindung wieder.

9.1.3.2 Suche nach Einträgen

Das LDAP SDK stellt eine Methode bereit, um eine Suche im Directory Server durchzuführen. Dabei muss zunächst ein LDAPSearchResult-Objekt erzeugt werden. In ihm wird das Ergebnis der Suche gespeichert.

```
LDAPSearchResult Result = new LDAPSearchResult();
```

Die eigentliche Suche wird über die Methode

```
Results=ld.search(base,scope,searchfilter,search_attri
butes,attrs_only );
```

durchgeführt. Dabei werden die folgenden Parameter übergeben

- **base**, gibt den BaseDN an, d.h. den Startpunkt im Verzeichnis ab dem gesucht werden soll, z.B. ou=Cruises,dc=awi-bremerhaven,dc=de, es wird ab dem Zweig ou=Cruises gesucht.
- **scope,** gibt die Reichweite der Suche an. Es gibt drei Möglichkeiten
  - LDAPv2.SCOPE_SUB, sucht ab dem angegebenen BaseDN und in allen Untereinträgen
  - LDAPv2.SCOPE_ONE, durchsucht nur die Zweige, die sich eine Stufe unter dem BaseDN befinden. Der BaseDN wird nicht durchsucht.
  - LDAPv2.SCOPE_BASE, die Suche wird nur in der angegebenen BaseDN durchgeführt.
- **searchfilter**, definiert den Suchfilter, das wonach gesucht werden soll, z.B. searchfilter=(&(cruiseship=Polarstern)(cruisebeginn=*2001)

Dieser Filter sucht nach allen Einträgen in denen das Attribut cruiseship den Wert Polarstern hat und der Wert des Attributes cruisebeginn mit 2001 endet.

- **search_attributes,** gibt an welche Attribute im Suchergebnis enthalten sein sollen, z.B. search_attributes ={„dn“, „cruiseid“}, im Suchergebnis sind nur die Attribute dn und cruiseid enthalten.
- **AttrsOnly,** gibt an das man entweder nur die Attributnamen oder die Attribute mit ihren Werten erhalten will.

9.1.3.3 Einträge hinzufügen

Um einen neuen Eintrag in den Directory Server zu schreiben, muss man zunächst festlegen, um welche Art von Eintrag es sich handelt und welche Attribute und Werte er beinhalten soll.

Mit

```
LDAPAttributeSet attrSet = new LDAPAttributeSet();
```

wird ein Objekt erzeugt, das die Attribute und die Werte des neuen Eintrags beschreibt. Dies Objekt ist zunächst leer. Über den Aufruf

```
LDAPAttribute attribute=new LDAPAttribute(„cruiseid“,
„ANT-XX/1“);
```

wird ein LDAPAttribute-Objekt erzeugt. Dabei wird der Name des Attributes und sein Wert übergeben. Dies wird für alle Attribute, die der Eintrag enthalten soll durchgeführt. Wichtig ist, dass das Attribut für die Objektklasse nicht vergessen wird. Im Fall eines Expeditionseintrages wäre das objectclass=awiCruise.

Jedes Attribut, das angelegt wird, wird dem LDAPAttributeSet hinzugefügt.

```
AttrSet.add(attribute);
```

Wenn alle Attribute angelegt und dem AttributeSet hinzugefügt wurden, wird ein LDAPEntry erzeugt. Dabei wird der zukünftige DN des Eintrages und das AttributeSet übergeben.

```
String DN =“cruiseid=ANT-XX/1,ou=Cruises,dc=awi-
bremerhaven,dc=de” ;
LDAPEntry new_entry = new LDAPEntry(DN, attrSet);
```

Durch die Methode

```
ld.add(new_entry);
```

wird der neue Eintrag schließlich in den Directory Server geschrieben.

9.1.3.4 Einträge modifizieren

Wenn ein bereits bestehender Eintrag geändert werden soll, können die folgenden Fälle auftreten.

Man möchte

- den Wert eines oder mehrerer Attribute ändern
- den Wert eines Attributs löschen
- einem mehrwertigen Attribut neue Werte hinzufügen
- ein neues Attribut hinzufügen
- ein Attribut entfernen.

Zunächst wird ein LDAPModificationSet erzeugt, das die zu ändernden Attribute und ihre Werte beinhalten soll.

```
LDAPModificationSet modSet=new LDAPModificationSet();
```

Auch hier wird wieder ein LDAPAttribut unter Angabe des Attributnamens und seines Wertes erzeugt.

```
LDAPAttribute attribute=new
LDAPAttribute(„cruiseship“, „Polarstern“);
```

Das Attribut wird dem LDAPModificationSet hinzugefügt.

```
modSet.add(LDAPModification.REPLACE, attibute);
```

Dabei wird die Art der Modifizierung übergeben. Folgende Werte sind dabei möglich

- REPLACE, der bereits vorhandene Wert eines Attributs wird durch den neuen Wert ersetzt.
- ADD, ein mehrwertiges Attribut erhält zu den bereits vorhanden Werten einen neuen zusätzlichen Wert.
- DELETE, das Attribut wird aus dem Eintrag entfernt.

Ein Attribut wird auch entfernt, wenn kein Wert für ein Attribut angegeben wird. Ein Attribut, das vorher noch nicht im Eintrag enthalten war, wird automatisch dem Eintrag hinzugefügt.

Durch die Methode

```
ld.modify(DN, modSet);
```

wird der übergebene Eintrag anhand des übergebenen LDAPModificationSets im Directory Server geändert.

9.1.3.5 Einträge löschen

Das Löschen eines Eintrages erfolgt durch den Aufruf der Methode

```
ld.delete(String DN);
```

Dabei wird der Eintrag mit dem übergebenen DN aus dem Directory Server gelöscht.

9.1.3.6 Authentifizierung

Die vorher beschriebenen Vorgänge Hinzufügen, Modifizieren und Löschen eines Eintrages erfordern in der Regel eine vorherige Authentifizierung des Benutzers.
Die Authentifizierung gegen den Directory Server kann einmal erfolgen durch separate Herstellung einer Verbindung mit dem Directory Server und einem späteren Aufruf der Methode

```
ld.authenticate(String DN, String password);
```

unter Übergabe des DNs des Benutzers und seines Passwortes.
Andererseits kann man gleich mit der Herstellung der Verbindung die Authentifizierung durchführen.

```
ld.connect(String host, int port, String DN, String
passwort);
```

9.2 Zusatzinformationen aus Verzeichnisssen

Um zusätzliche Informationen zu den einzelnen Expeditionsfahrten anzeigen zu können, werden Dateiverzeichnisse auf dem Server „e-net“ durchsucht.
Dabei handelt es sich um Verzeichnisse, die Karten, Kurzfassungen, Pressemitteilungen und Wochenberichte zu den einzelnen Expeditionen enthalten. Dazu wird in den einzelnen Methoden, die die entsprechenden Verzeichnisse durchsuchen ein Filter definiert, der nur die Dateien mit der angegebenen Dateiendung betrachtet. Für Karten ist das z.B. .png. Die Pressemitteilungen und Wochenberichte liegen als HTML-Dateien vor.

9.3 Authentifizierung und Autorisierung

Jeder Benutzer der sich einloggen will, um den Administrationsbereich zu nutzen, muss sich zunächst gegenüber dem Directory Server authentifizieren. Dies geschieht, indem er seinen Emailbenutzernamen und sein Emailpasswort eingibt. Zunächst wird geprüft,

ob es den eingegebenen Benutzer gibt. Anschließend wird geprüft, ob das eingegebene Passwort mit dem im Personeneintrag des eingegebenen Benutzers übereinstimmt.
Wenn sich eine Person bei der Anwendung einloggt, muss geprüft werden, ob sie die nötigen Rechte besitzt, um eine Administratorrolle einzunehmen.
Die Autorisierung erfolgt durch das Prüfen auf Zugehörigkeit zu der im Directory Server eingetragenen Gruppe „CruisePortal“. Nur Personen, die einen Eintrag „uniquemember“ in dieser Gruppe haben, haben die Berechtigung als Administrator für die Expeditionsanwendung. Alle anderen Personen, die sich einloggen, werden automatisch als normaler Benutzer gesetzt.

9.4 Sessions

Das Hypertext Transfer Protocol (HTTP) ist zustands- und verbindungslos, d.h. es kann keine Verbindung zwischen einem Benutzer und Server über mehrere Anfragen und Antworten speichern.
Mit Hilfe von Sessions können Daten über mehrere Anfragen und Antworten gespeichert werden. Die Daten werden dabei serverseitig abgelegt. Auf Clientseite wird lediglich die SessionID in einem Cookie gespeichert. Bei jeder Anfrage wird der Cookie, der die SessionID enthält, vom Browser an das Servlet übermittelt. Wird vom Server bzw. vom Servlet eine Session erzeugt, ist diese solange nicht gültig, bis der Client sie mit einer Anfrage zurücksendet. In diesem Moment hat der Client die Session akzeptiert und die Session kann benutzt werden.
Eine Session wird durch die Schnittstelle HttpSession in der Servlet API repräsentiert.
In der Anwendung werden Sessions dazu genutzt, um verschiedene Objekte während der Dauer der Nutzung zu speichern. Das sind z.B. Informationen über die gewählte Spracheinstellung oder den Modus des aktuellen Benutzers
Wenn die Anwendung gestartet wird, wird automatisch eine neue Session erzeugt.
Mit der Anweisung

```
HttpSession session =request.getSession(boolean);
```

wird ein neues Session-Objekt erzeugt. Der boolean-Wert gibt dabei an, ob eine neue Session erzeugt werden soll, wenn noch keine vorhanden ist. Außerdem kann man so auf eine bereits vorhandene Session zugreifen. Anschließend kann die maximale Dauer, die die Session existieren soll, gesetzt werden. Dabei wird die Dauer in Sekunden angegeben.

```
session.setMaxInactiveInterval (28800);
```

Durch den Aufruf der Methode

```
session.invalidate();
```

wird die bestehende Session zerstört.

Man hat die Möglichkeit in der Session verschiedene Objekte zu speichern. Dies geschieht dadurch, das man sie in die Session einbindet. Über die Methode

```
session.setAttribute(Attributname, Attributwert);
```

kann man in der Session ein Objekt speichern und ihm einen Wert zuweisen.

Über die Methode

```
session.getAttribute(Attributname);
```

kann auf das in der Session gespeicherte Objekt zugegriffen und so dessen Wert abfragt werden.

In der Anwendung werden verschiedene Informationen in der Session gespeichert.

Die in der Session gespeicherten Session-Objekte sind im einzelnen

- **„is_Authenticated“**, gibt an, ob der Benutzer authentifiziert ist oder nicht. Mögliche Werte sind true und false.
- **„mode“**, speichert den Benutzermodus, in dem sich der aktuelle Benutzer befindet. Möglich sind hier die Werte „representative“ für die Rolle als Administrator und „user“ für den normalen Benutzer.
- **„PersonToEdit“**, beinhaltet ein Objekt der Klasse „Person“, welches die Daten des Directory Server Eintrages des aktuellen Benutzers enthält.
- **„locale“**, enthält ein Objekt der Klasse „Locale“, in dem der aktuelle Sprach- und Ländercode abgespeichert wird.

9.5 Properties-Dateien

Es werden Propoerties-Dateien eingesetzt, um die Anwendung dynamisch zu halten. Sie soll leicht zu warten und zu ändern sein.

Properties-Dateien sind Textdateien, die eine Schlüssel-Wert-Collection beinhalten.

Über den „Schlüssel“ kann auf die Werte zugegriffen und diese ausgelesen werden. Beispiel für eine Schlüssel-Wert-Zuweisung innerhalb einer Properties-Datei.

```
HOST   = e-net.awi-bremerhaven.de
```

Schlüssel Wert

Die Properties-Dateien werden in der Anwendung dazu verwendet, um Konfigurationsdaten, Internationalisierungsvokabular und Bezeichnungen für die HTML-Ausgaben zu speichern. Die folgenden Properties-Dateien wurden implementiert

- § expedition.properties, dient als Konfigurationsdatei und beinhaltet Angaben wie z.B. über den Hostnamen und den Port des Directory Servers, Pfadangaben.
- § expeditionUI_de_DE.properties, enthält das Vokabular für die HTML-Ausgaben mit den deutschen Bezeichungen
- § expeditionUI_en_EN.properties, enthält das Vokabular für die HTML-Ausgaben mit den englischen Bezeichnungen
- § expedition_Ship_Name_des_Schiffes.properties, für jedes Schiff, das in die Anwendung integriert wird, wird eine eigene Datei angelegt, die schiffsspezifische Konfigurationsangaben enthält.

Der Einsatz von Properties-Dateien bietet einige Vorteile. Die Textausgaben der Anwendung sind nicht statisch im Java-Quellcode verankert. So können die Ausgaben leicht verändert werden, indem einfach die gewünschten Änderungen in der jeweiligen Properties-Datei vorgenommen werden. Analog verhält es sich mit den Konfigurationsangaben. Wenn sich der Hostname oder Port des Directory Servers ändert, reicht es aus den entsprechenden Wert in der Properties-Datei zu ändern. Es entfällt der Aufwand den gesamten Quellcode nach zu ändernden Hostnamen zu durchsuchen.

9.6 Javascript

Unter der Verwendung von Javascript wird die dynamische Wertebelegung der Pulldown-Menüs realisiert.

9.7 Programmierkonzepte

Während der Programmierung und dem Entwurf der Anwendung wurden einige Konzepte beachtet, um spätere Wartung und Erweiterungen des Programms zu erleichtern.

Um das Verständnis und die Nachvollziehbarkeit des Quellcodes für andere Programmierer zu fördern, bietet sich die Dokumentation in Form von Kommentaren

an. Im Javacode der einzelnen Klassen wurden die Methoden und Klassen mit Kommentaren versehen.
Zusätzlich beinhalten alle Java-Klassen Javadoc-Kommentare aus denen mit Hilfe des Tools Javadoc aus dem JDK die Dokumentation der Klassen und Pakete generiert wird.
Es wurden möglichst eindeutige Bezeichnungen für Klassen, Attribute und Methoden gewählt. Dabei wurden gängige Konventionen bei der Bezeichnung beachtet

9.8 Programmablauf

Im folgenden wird der Programmablauf, gegliedert in den öffentlichen und den nichtöffentlichen Bereich der Anwendung, beispielhaft beschrieben.

9.8.1 öffentlicher Bereich

Die Einstiegsseite der Anwendung zeigt den dynamisch generierten Zeitplan. Als Standard sind zunächst das Forschungsschiff „R.V. Polarstern“ und das aktuelle Jahr ausgewählt. Der Kopf der Anwendung zeigt einen Header mit Links mit deren Hilfe der Benutzer durch das Angebot navigieren kann. Dieser Header ist immer gleichbleibend und wird auf jeder Seite der Anwendung ausgegeben. Über der generierten Ausgabe des Zeitplans werden Informationen über das jeweilige ausgewählte Forschungsschiff ausgeben.

Abbildung 36: Einstiegsseite der Anwendung - Zeitplan

Zu den chronologisch aufgelisteten Expeditionsfahrten im Zeitplan werden in Links weiterführende Zusatzinformationen aufgeführt.

Für jede Expedition werden, wenn vorhanden, Karten mit dem Fahrtverlauf, Wochenberichte, Kurzfassungen und Pressemitteilungen aufgeführt. Dazu gehören auch eine Auflistung der zu der Expedition gehörenden AWI Publikationen, PS Abstracts und CruiseReports. Des weiteren hat der Benutzer die Möglichkeit über einen Link zu einer Abfragemaske von meteorologischen Daten zu gelangen. Dabei kann er verschiedene Kriterien selektieren, nach denen eine Datenbankabfrage einer Sybase DB

durchgeführt wird. Das Ergebnis der Abfrage wird in einer ASCII(American Standard Code for Information Interchange)-Datei ausgegeben.

Der dynamisch generierte Zeitplan kann als PDF-Datei exportiert werden. Die PDF-Datei wird dabei dynamisch auf Grundlage der gleichen Daten erzeugt. Die Anwendung kann über eine Sprachauswahl in deutsch oder englisch angezeigt werden.

Der Benutzer kann sich zu jeder im Zeitplan aufgeführten Expedition eine Detailansicht ansehen.

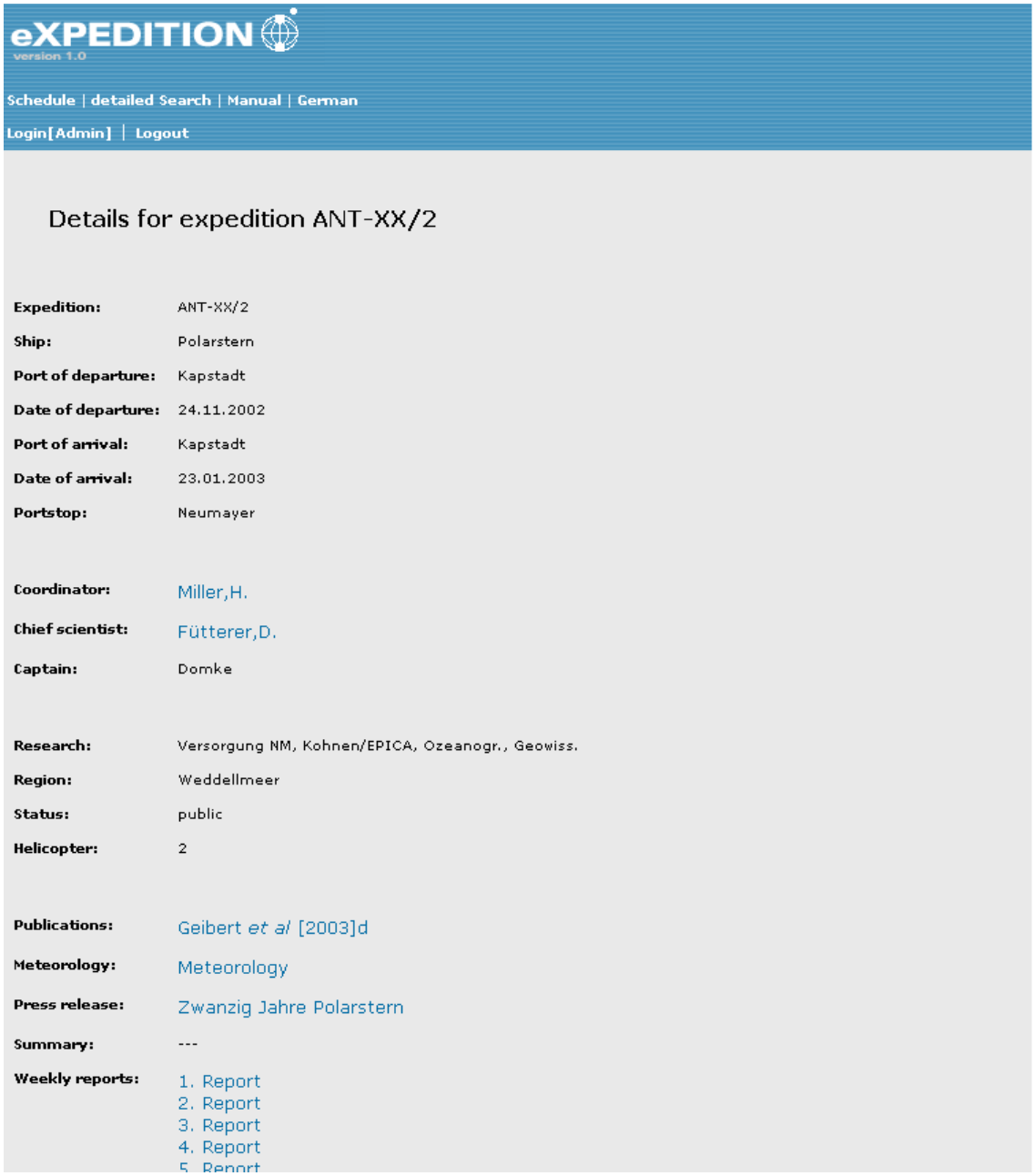

Abbildung 37: Detailansicht eines Expeditionseintrages

In der Detailansicht werden die Einzelheiten der Expeditionsfahrt unter Angabe von z.B. Abfahrts-/Ankunftshafen, Abfahrts-/Ankunftsdatum, Fahrtkoordinator, Fahrtleiter, Kapitän, Forschungsgebiet und Forschungsthemata ausgegeben. Daneben werden auch die schon im Zeitplan erwähnten zusätzlichen Informationen aufgeführt.

Der Benutzer kann über Pulldown-Menüs das ausgewählte Forschungsschiff und das Expeditionsjahr für die er den Zeitplan anzeigen lassen möchte, ändern. Momentan umfasst es die beiden Forschungsschiffe „R.V. Polarstern“ und die „R.V. Heincke“. Für beide Schiffe werden unterschiedliche Arten von Informationen ausgegeben. Für die Heincke werden beispielsweise keine Publikationen angezeigt. Des weiteren kann der Benutzer sich Informationen über die anderen Forschungseinrichtungen anzeigen lassen. Über ein Pulldown-Menü kann der Typ der Forschungseinrichtung auswählt werden. Möglich sind im Moment die Auswahl der Einrichtungen „Research Vessel“, „Aircraft“, „Land-based station“ und „Ocean-based station“. Entsprechend der Auswahl werden die für die Forschungseinrichtung vorhandenen Namen in einem Pulldown-Menü aufgelistet. Zu jeder Auswahl kann der Benutzer sich die spezifischen Informationen ansehen.

Die Anwendung umfasst des weiteren eine Suchfunktion mit der der eingetragene Bestand an Expeditionsfahrten nach bestimmten Kriterien durchsucht werden kann. Über eine Suchmaske kann der Benutzer so z.B. den Bestand nach allen Fahrten des Forschungsschiffes „R.V. Polarstern“ aus dem Jahr 2000 die in Bremerhaven gestartet sind durchsuchen.

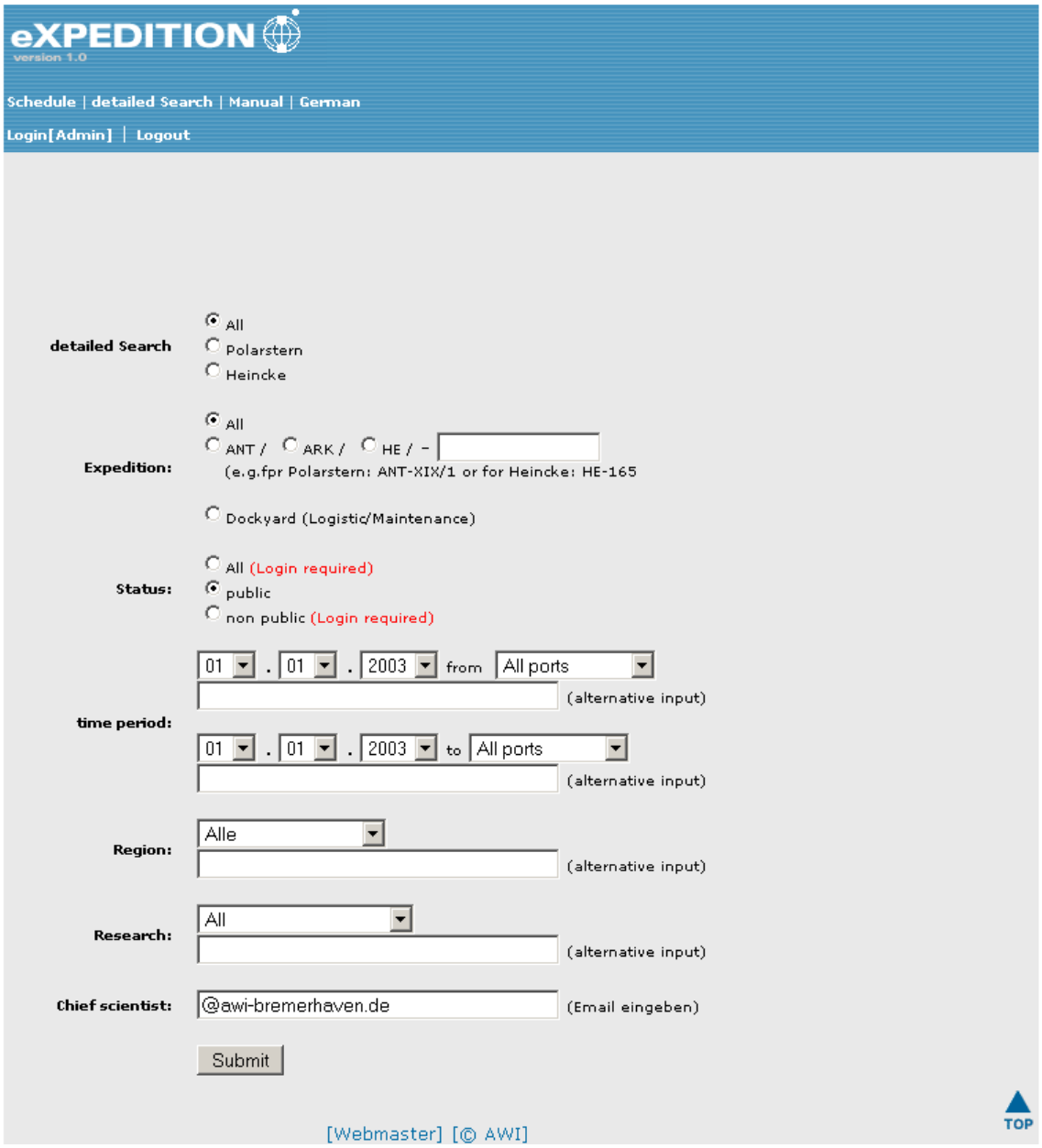

Abbildung 38: Suchmaske

Das Suchergebnis listet die gefundenen Expeditionen chronologisch auf. Dabei wird auch ein Verweis auf die Detailansicht ausgegeben. Außerdem hat der Benutzer die Möglichkeit über einen Link die gewählte Fahrt zu modifizieren. Dabei wird zunächst geprüft, ob der Benutzer bereits eingeloggt ist und die entsprechenden Rechte als Administrator besitzt. Ist dies der Fall, wird ihm die Maske zum Modifizieren des Expeditionseintrages angezeigt. Andernfalls wird die Loginmaske angezeigt und der Benutzer muss sich erst einloggen.

9.8.2 Nichtöffentlicher Bereich

Die Anwendung umfasst neben dem für alle Benutzer zugänglichen öffentlichen Bereich auch einen nichtöffentlichen Bereich. Der nichtöffentliche Bereich ist nur Benutzern zugänglich, die eine Administratorrolle für die Anwendung einnehmen.

Um den Administrationsbereich zu nutzen, muss sich der Benutzer über eine Loginmaske unter Angabe seines Emailbenutzernamens und Passwortes einloggen.

Abbildung 39: Loginmaske

Dabei wird er zunächst authentifiziert. Anschließend wird geprüft, ob der Benutzer die nötige Autorisation besitzt. Ist der Benutzer berechtigt, gelangt er in den nichtöffentlichen Bereich. Ansonsten erhält er eine entsprechende Meldung, dass er nicht über die erforderlichen Rechte verfügt. Die Startseite des Administrationsbereichs zeigt eine Übersicht über alle eingetragenen Expeditionsfahrten in chronologischer Ordnung an.

Abbildung 40: Übersicht nichtöffentlicher Bereich

Jeder aufgeführte Expeditionseintrag kann selektiert, gelöscht und modifiziert werden. Je nach Auswahl wird die entsprechende Maske angezeigt.

Des weiteren hat der Benutzer die Möglichkeit über eine Eingabemaske neue Expeditionsfahrten einzutragen.

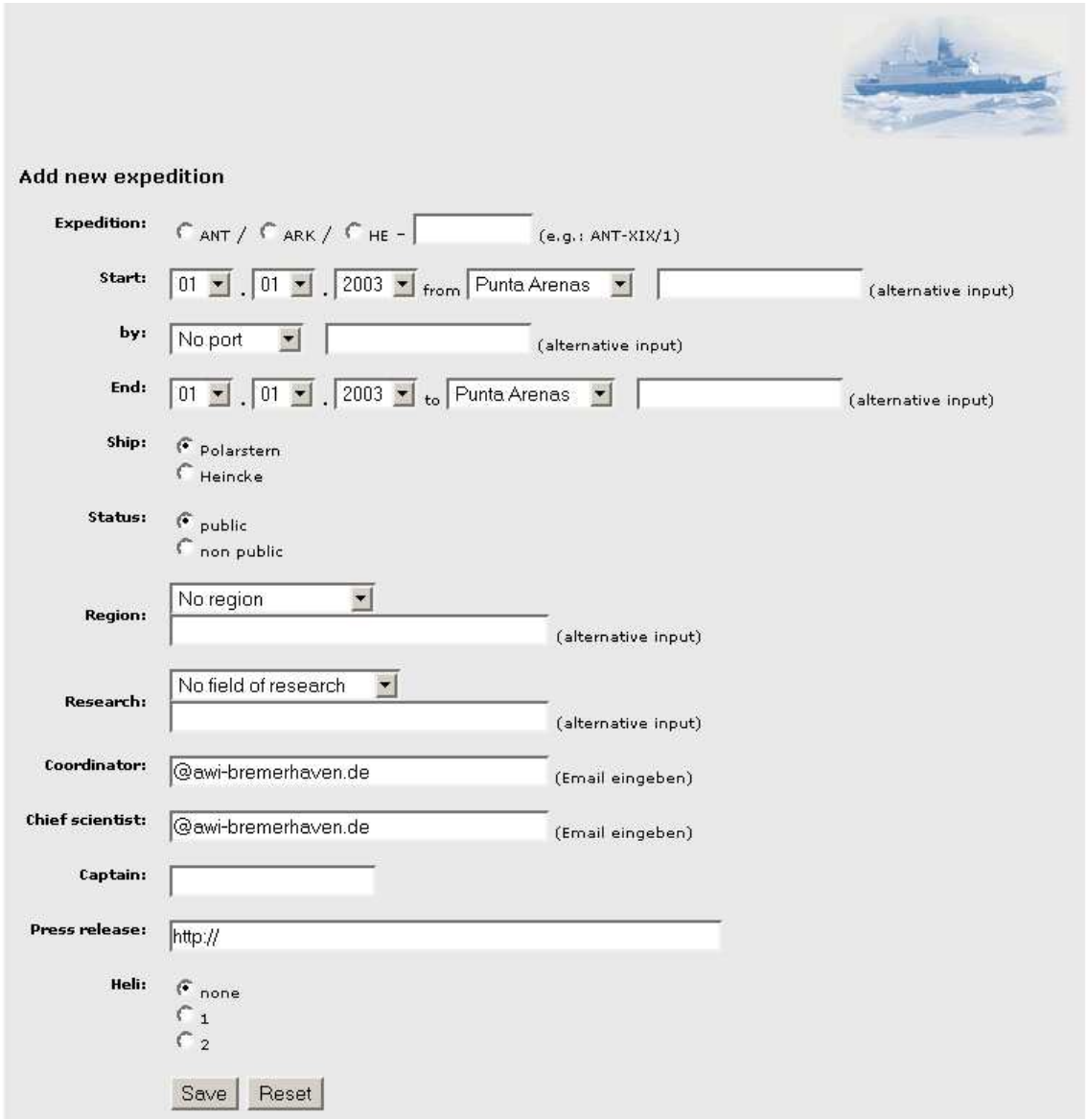

Abbildung 41: Maske zur Eintragung neuer Expeditionen

10 Fazit und Perspektiven

Der erstellte Prototyp des Expeditionsportals stellt eine Basis dar, die gut erweiterbar ist. Durch die modulare Programmierung und den Einsatz der Properties-Dateien ist das Programm dynamisch und kann leicht um neue Komponenten und Funktionalitäten erweitert werden. Es möglich leicht neue Forschungseinrichtungen, wie z.B. ein neues Forschungsschiff oder eine Forschungsstation, ohne größeren Programmieraufwand in die Anwendung zu integrieren. Änderungen an den Textausgaben können schnell und einfach vorgenommen werden. Die Anwendung ist problemlos auf andere Server (Webserver, Directory Server) portierbar, da die Konfigurationen in den Konfigurationsdateien vorgenommen werden.

Anhangverzeichnis

Der Arbeit beigefügt ist eine CD. Diese CD enthält neben der Arbeit im Word-Format auch den Java-Quellcode der entwickelten Anwendung.

Glossar

ASP

Active Server Pages

Technik des IIS zur Gestaltung dynamisierter HTML-Seiten. ASP ist also eine Alternative zur Verwendung von CGI und SSI. Es ermöglicht die Einbindung von VBScript, JavaScript und sogenannter Active Data Objects (ActiveX- bzw. OLE-Automation-Server-DLLs) auf der Server-Seite. Bevor eine Seite verschickt wird, werden die darin enthaltenen Scripts ausgeführt. Beim Empfänger kommt aber nur reiner HTML-Text an, die Scripts selbst sind nicht mehr enthalten. über die Scripts können z.B. Datenbankanschlüsse hergestellt oder Daten aus Eingabeformularen verrechnet werden.[12]

Client

PC-Netze sind in den meisten Fällen dadurch gekennzeichnet, dass sie zwei Arten von Endgeräten beinhalten: Clients und Server. Die Clients sind die Arbeitsplatzrechner und nutzen die von den Servern angebotenen Dienstleistungen. Dazu besitzen sie eine Requester-Komponente, die mit dem Netzwerkbetriebssystem auf den Servern kooperiert.

Der oder die Server können zur Organisation des Netzes über die Kontrolle der logischen Betriebsmittel herangezogen werden. Im Gegensatz zu Client-Server-Architekturen stehen Peer-to-Peer-Netze, bei denen die Betriebsmittel von der Gesamtheit der Benutzer bereitgestellt und kontrolliert werden. In ihnen gibt es keinen Server.[13]

Client-Server-Architektur

Die Client/Server-Architektur bezeichnet ein Systemdesign bei dem die Verarbeitung einer Anwendung in zwei separate Teile aufgespaltet wird. Ein Teil läuft auf dem Server (Backend-Komponente), der andere Teil auf einer Workstation (Client oder Front-End). Beide Teile werden über Netzwerke zu einem System zusammengefügt. Der Client gibt auf dem Server die Bearbeitung von Daten in Auftrag und nimmt die Leistungen des Servers in Anspruch. Im Gegensatz zu Host-basierten Architekturen

[12] vgl. [Seicom, 2003]
[13] vgl. [Siemens, 2003]

sind die Server heute nicht mehr mit der gesamten Datenverarbeitung beschäftigt, sondern geben die Daten zur weiteren Aufbereitung an den Client zurück. Während auch viele Kommunikationsprotokolle mit Client-Server-Prozeduren arbeiten, bezieht sich die Bezeichnung C/S-Architektur aber meist auf die Verteilung von Anwendungen.
Die Arbeitsteilung zwischen Client und Server kann auf sehr unterschiedlichen Ebenen einer Systemanwendung erfolgen. Um die Aufgabentrennung besser beschreiben zu können, wird oft versucht, verteilte Anwendungen in ein Schichtenmodell zu unterteilen. Im Gegensatz zum OSI-Modell gibt es aber noch kein standardisiertes Modell für verteilte Anwendungen.[14]

Cookies

Cookies sind kleine Textdateien in Browsern, die ein Server an einen Browser schickt. Darin wird das Verhalten des Nutzers registriert: die Passwörter, persönliche Daten des Nutzers, welche Seiten er am häufigsten aufruft. Die Dateien werden gespeichert und beim nächsten Aufruf des Servers wieder abgerufen. Dies erleichtert Login-Prozeduren, kann aber auch benutzt werden, um Nutzungsgewohnheiten eines Besuchers zu protokollieren und Benutzerprofile zu erstellen.[15]

FTP-Protokoll

Das File-Transfer-Protokoll (FTP) dient dem Dateitransfer zwischen verschiedenen Systemen und der einfachen Dateihandhabung. FTP basiert auf dem Transportprotokoll TCP und kennt sowohl die Übertragung zeichencodierter Information als auch von Binärdaten. In beiden Fällen muss der Benutzer eine Möglichkeit besitzen zu spezifizieren, in welcher Form die Daten auf dem jeweiligen Zielsystem abzulegen sind. Die Dateiübertragung wird vom lokalen System aus gesteuert, die Zugangsberechtigung für das Zielsystem wird für den Verbindungsaufbau mittels User-Identifikation und Passwort überprüft.
Will ein Client mit dem Server kommunizieren, baut der Benutzer über den Interpreter im Client eine Verbindung zum Interpreter im Server auf. Über diese Steuerverbindung kommunizieren Client und Server über einen Satz festgelegter Kommandos. Es gibt

[14] vgl. [Seicom, 2003]
[15] vgl. [Siemens, 2003]

mehrere Kommandosätze für die Zugriffssteuerung, die Befehle für den Datentransfer und die FTP-Services. Zum Austausch der Nutzdaten baut der Server eine zweite Verbindung zum Client auf, und zwar wird diese Verbindung von dem Datentransfer-Prozess (DTP) gesteuert. Über diese Verbindung werden die Nutzdaten übertragen. Beim Verbindungsabbau bestätigt der Server-Interpreter das Ende des Datentransfers über die Steuerverbindung an den Interpreter des Clients.[16]

HTTP

hypertext transport protocol

HTTP ist ein allgemeines, statusloses, objektorientiertes Protokoll zur Datenübertragung im Rahmen des World Wide Web (WWW). Das HTTP-Protokoll ist ein einfaches Protokoll, das vom Prinzip her dem Gopher-Protokoll ähnelt. Es beschreibt einen definierten Satz von Nachrichten und Antworten, mit denen ein Client und ein Server während einer HTML-Sitzung kommunizieren. Jede Anfrage eines Web-Browsers an einen Web-Server nach einem neuen Dokument stellt eine neue Verbindung dar. Das HTTP-Protokoll dient der Adressierung der Objekte über URL, es wickelt die Interaktion zwischen Client und Server ab und sorgt für die Anpassung der Formate zwischen Client und Server. Das Serverprogramm zu HTTP ist httpd. Es beantwortet Anfragen von WWW-Clients.[17]

PDF

Portable Document Format

Hypertext-fähiges Dateiformat zum Austausch von fertig formatierten Dokumenten. PDF wurde von Adobe auf Basis von PostScript-Sprache entwickelt und um Hyperlinks, Datenkompression und Verschlüsselung erweitert. PDF-Dateien können aus allen druckfähigen Programmen heraus über frei verfügbare PDF-Druck-Spooler erzeugt werden. Eine Weiterverarbeitung zum Hypertext ist aber nur mit speziellen Werkzeugen möglich (und meist auch nicht erwünscht, da aufwendig). PDF eignet sich insbesondere für die elektronische Publikation und Verteilung vorhandener Papierdokumentation, die mit ihrem kompletten Layout erhalten bleiben soll. WWW-Browser unterstützen PDF über Plug-ins (Adobe Acrobat oder GhostView).

[16] vgl. [Siemens, 2003]
[17] vgl. [Siemens, 2003]

Eine FDF genannte Erweiterung erlaubt auch den Einsatz von PDF für Online-Formulare. Diese Technik ist im Internet aber kaum verbreitet.[18]

PHP

PHP HyperText Preprocessor

PHP ist eine Scriptsprache, die speziell für das Internet entwickelt wurde. Wie auch ASP kann PHP direkt in Kombination mit HTML genutzt werden. PHP bietet einen großen Funktionsumfang, der auch die schon von Perl bekannten Zeichenkettenfunktionen umfasst. Eine große Anzahl von verschiedenen Datenbanken werden unterstützt - z.B. MySQL, MSSQL, Oracle, und viele mehr.
Das sogenannte LAMP-System kombiniert Linux, Apache, MySQL, PHP und ist mitlerweile sehr verbreitet, insbesondere da alle Komponenten Open Source sind.[19]

Server

Meist organisiert man PC-Netze in einer zweistufigen Systemhierarchie bestehend aus Dienstanbietern (Servern) und Dienstbenutzern (Clients). Die Server realisieren funktionale und infrastrukturelle Netzdienste, das heißt, sie bieten nicht nur den Clients Funktionen an, sondern ermöglichen auch die Netzadministration. Server sind üblicherweise die stärksten und am besten ausgebauten Rechner im Netz. Sie besitzen große Festplattenkapazitäten, eine schnelle CPU und, wenn möglich, darüber hinaus noch unterschiedliche Coprozessoren. Server wohnen in ausbaufähigen Gehäusen, die auch eine spätere Nachrüstung mit neuen Speichermedien ermöglichen. Neben den üblichen Externspeichern auf Diskettenbasis können Server auch Bandlaufwerke, Bernoulli-Boxen, Wechselplatten oder CDE-Laufwerke besitzen, um einer ihrer wichtigsten Funktionen, dem automatischen Backup, angemessen nachgehen zu können. Manche Teile der Server können ausfallsicher ausgelegt sein. So unterstützen die PC-LAN-Betriebssysteme NetWare und LAN-Manager gespiegelte Festplatten oder die doppelte Ausführung der Plattensubsysteme sowie unterbrechungsfreie Stromversorgungen, UVS.
Das wichtigste Programm auf dem Server ist das Netzwerkbetriebssystem. Diese Software erlaubt die gemeinsame geordnete Benutzung von Betriebsmitteln (mindestens

[18] vgl. [Seicom, 2003]
[19] vgl. [Seicom, 2003]

File- und Print Sharing) und die Installationen von Software für die Implementierung zusätzlicher Dienste. Je nach Netzwerkbetriebssystem brauchen die Server kein weiteres eigenes Betriebssystem. Allerdings sind auch Netzwerkbetriebssysteme bekannt, die im Status eines Anwendungsprogramms unter einem anderen Betriebssystem laufen. Die Client-Server-Architektur ist auch auf andere Rechnertypen übertragbar. So kann ein Großrechner Datei-Server und Print-Server für eine Menge von LAN-Servern werden, die dann in die Rolle von Clients schlüpfen. So können die Vorteile einer Großrechnerumgebung wie z.B. hohe Sicherheit auch für alle LAN-Benutzer bereitgestellt werden.[20]

TCP/IP-Protokolle

transmission control protocol/internet protocol

Die TCP/IP-Protokolle wurden schon vor ca. 30 Jahren von der Research Project Agency (DARPA) des US-Verteidigungsministeriums (DoD) mit Unterstützung des National Bureau of Standards (NBS) entwickelt. Ziel war die Schaffung möglichst Code-kompakter Protokolle für die IMPs des ARPAnet. Danach hat sich lange Zeit niemand mehr für TCP/IP interessiert, da man hoffte, auf Basis von OSI-Protokollen einen Verbund heterogener Systeme realisieren zu können. Diese Hoffnung trog und die mittlerweile auf dem Unix-Sektor weiterentwickelten Protokolle wurden schnell für die Vernetzung unterschiedlichster Systeme populär. Wegen des vergleichsweise geringen Implementierungs- und Platzaufwandes wurden sie auch gerne für PCs verwendet. Heute findet man kein professionelles System, das nicht mit TCP/IP-Protokollen ausgestattet werden kann. Somit ist die DoD-Protokollfamilie in vielen gemischten Umgebungen heute die einzige Möglichkeit der Kommunikation. Der relativ geringe Aufwand hat natürlich auch seinen Preis: in diesem Falle in der Funktionenvielfalt und in der Leistung z.B. beim Filetransfer. Die Entwicklung der TCP-Protokolle wurde wesentlich durch die Erfahrungen im DARPA Internet (früher ARPAnet) beeinflusst, ebenso durch eine Reihe von Protokollspezifikationen und so genannte »Requests For Comments« (RFC). Das amerikanische Verteidigungsministerium verwaltet die Spezifikationen der TCP/IP-Protokollfamilie. Alle Protokolle aus dem TCP/IP-Umfeld unterstützen direkt die Funktionalität der Vermittlungsschicht und der Transportschicht sowie verschiedene Dienste, die auf den TCP/IP-Protokollen aufbauen. Die

[20] vgl. [Siemens, 2003]

bekanntesten Protokolle für die Vermittlungsschicht sind das IP-Protokoll und das datagrammorientierte UDP-Protokoll sowie für das Routing das OSPF-Protokoll und das RIP-Protokoll. Die Transportschicht wird unterstützt von dem TCP-Protokoll, auf das Dienste wie Telnet, FTP, SMTP, r-utilities, X-Window oder SNMP aufsetzen. [21]

UML

Die Unified Modelling Language (UML) ist eine Standardsprache für die objektorientierte Modellierung von Softwareentwürfen. Sie dient dem Visualisieren, Spezifizieren, Konstruieren und Dokumentieren der Konstrukte eines softwareintensiven Systems. Die UML eignet sich zum Modellieren von Systemen angefangen von unternehmensinternen Informationssystemen über verteilte webbasierte Anwendungen bis hin zu eingebetteten harten Echtzeitsystemen. Sie ist eine sehr ausdrucksvolle Sprache, die alle für die Entwicklung und den Einsatz solcher System notwendigen Sichten mit einbezieht. Trotz ihrer Ausdrucksstärke ist sie leicht zu verstehen und anzuwenden. Die UML beschreibt keine Methode. Sie ist nur eine Sprache, also bloß ein Teil einer Methode zur Softwareentwicklung.

[21] vgl. [Siemens, 2003]

Literaturverzeichnis

[AWI, 2003] Alfred-Wegener-Institut für Polar- und Meeresforschung Homepage, 2003
http://www.awi-bremerhaven.de

[AWIReport, 2002] „Das AWI in Jahren 2000 und 2001“ , Stiftung Alfred-Wegener-Institut für Polar- und Meeresforschung, Bremerhaven, 2002

[Balzert, 1999] Balzert, Heide, „Lehrbuch der Objektmodellierung – Analyse und Entwurf“, Spektrum, Akadem. Verlag, Heidelberg, Berlin 1999

[HelBalzert, 1996] Balzert, Helmut, „Lehrbuch der Software-Technik: Software-Entwicklung“, Spektrum, Akadem. Verlag, Heidelberg, Berlin 1996

[BraSchWab, 2001] Brandner, Matthias/ Schmidt, Stefan/ Wabnitz, Birgit und Detlef, „JavaServer Pages und Servlets“, Data Becker, Düsseldorf, 2001

[Duden, 2003] „Duden – die deutsche Rechtschreibung“, herausgegeben von der Dudenredaktion, Dudenverlag, 22. Auflage, Mannheim, 2003

[Hanke, 2002] Hanke, Johann-Christian, „Word für Studenten“, 1. Auflage, Knowware Verlag, 2002

[Horton, 1999] Ivor Horton „Beginning Java 2“, Wrox Press, Birmingham 1999

[HowSmiGor, 1999] Howes, Timothy/ Smith, Mark/ Good, Gordon, „Understanding and Deploying LDAP Directory Services“, Macmillan Computer Publishing, USA, 1999

[LDAPsdk, 2003] „Netscape Directory SDK 4.0 for Java Programmer’s Guide“, Sun Produktdokumentation, docs.sun.com, letzter Abruf am

07.07.2003 http://docs-pdf.sun.com/816-6402-10/816-6402-10.pdf

[MiddSing, 1999] Middendorf, Stefan/ Singer, Reiner, „Java Programmierhandbuch und Referenz“, 2. überarbeitete und erweiterte Auflage, dpunkt.verlag, Heidelberg, 1999

[PDFlib, 2002] Merz, Thomas, „Reference Manual PDFlib“, München, 2002 htttp://www.pdflib.com

[Portal, 2001] Schweizer, Karin, „Portal total“, IBM, 2001 http://www-5.ibm.com/de/software/enews/essay/2001-02-essay-portale.pdf, letzter Abruf am 18.08.2003

[Rossig, 2001] Rossig, Wolfram/ Prätsch, Joachim, „Wissenschaftliche Arbeiten – Ein Leitfaden für Haus-, Seminar-, Examens- und Diplomarbeiten sowie Präsentationen“, 3.Auflage, Wolfdruck Verlag Bremen , 2001

[Scholz, 2001] Scholz, Dieter , „Diplomarbeiten normgerecht verfassen – Schreibtipps zur Gestaltung von Studien-, Diplom- und Doktorarbeiten“, 1.Auflage, Vogel Verlag, Würzburg, 2001

[Seicom, 2003] Online-Lexikon, Seicom-Muc.de, 2003 http://www.seicom-muc.de/db_nav/fkt/show_nav.php4?mid=65 letzter Abruf am 28.07.2003

[Siemes, 2003] „Das Siemens Online Lexikon“, Siemens Information and Communications Deutschland, letzter Abruf am 28.07.2003 http://w3.siemens.de/solutionprovider/_online_lexikon/

[Sun, 2003] „Sun online documentation“, 2003, http://java.sun.com/ letzter Abruf am 16.08.2003

[WelDah, 2000] Weltman, Rob/ Dahbura, Tony, „LDAP programming with Java“, Addison Wesley, 2000

Versicherung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen sind, sind als solche unter Angabe der Quellen kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Bremerhaven, den 29.09.2003